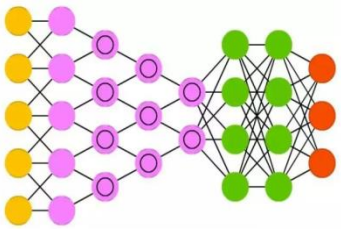
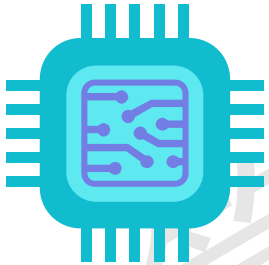
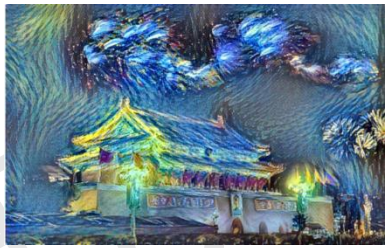
智能计算系统

第三章 深度学习

中国科学院计算技术研究所
陈云霁 研究员
cyj@ict.ac.cn

Driving Example

输入 → 建模 → 实现 → 运行 → 输出

神经网络基础

深度学习

上一章学习了神经网络的基本知识，多层感知机的正反向计算过程，以及基础优化方法。

本章通过分析经典深度学习算法，学习将基础神经网络应用到实际场景，并逐步优化实现工业级应用的过程。

让机器更好的理解和服务人类

图像信息

任务：理解图像内容

方法：卷积神经网络

序列信息

任务：理解语音/文字/视频

方法：循环神经网络

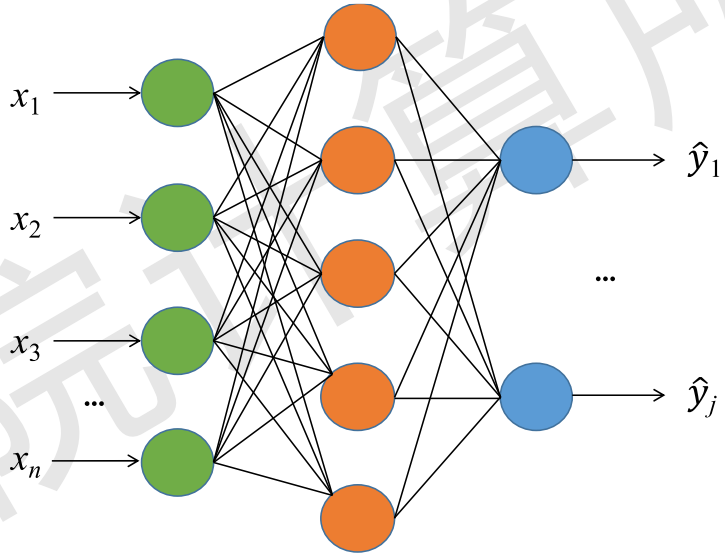
提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

一个例子

- 计算机视觉

输入图像

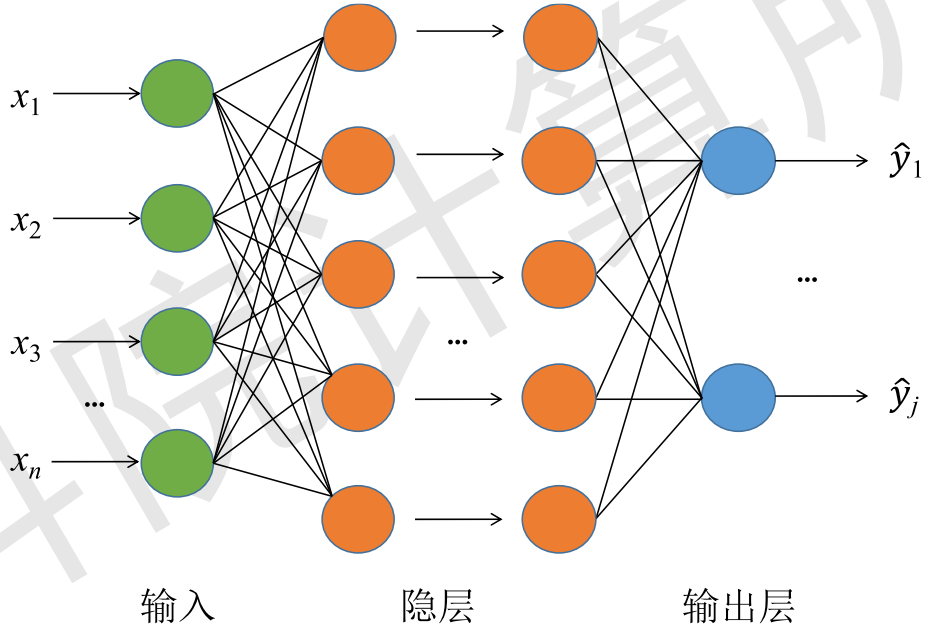


- 输入图像大小为 32x32，输入数据量为 32x32x3 = 3072
- 隐层神经元个数为 100，第一层权值数量为 3072 x100 = 307200

一个例子

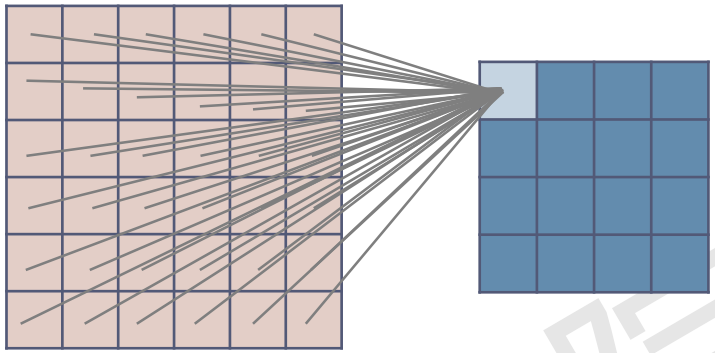
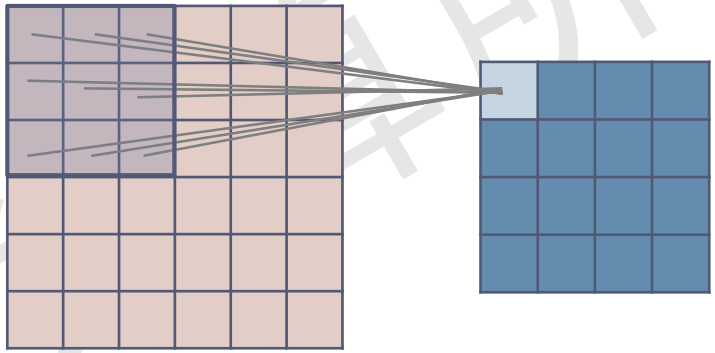
- 实际场景中，往往需要更大的输入图像以及更深的网络结构。

输入图像



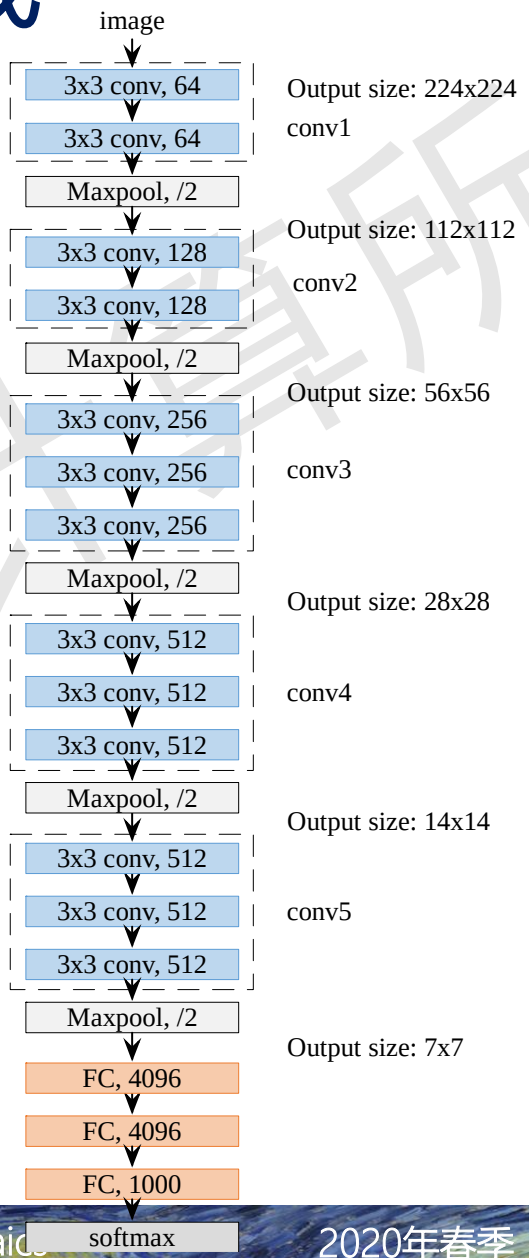
- 输入图像大小为 1024x1024，第一层隐层神经元个数为 1000
- 第一层权重数量级为 10^9，过多的参数会导致**过拟合**
- **卷积神经网络**可以有效减少权重数量

卷积神经网络（CNN）

	全连接	卷积
局部连接		
权重共享	所有神经元之间的连接都使用不同权重。	输出层神经元共用同一组权重， 进一步减少权重数量。
权重数量	$w_i \times h_i \times w_o \times h_o$	$f \times f$

CNN组成

- VGG16
  - 卷积层（conv）
  - 池化层（max pool）
  - 全连接层 （FC）
  - Softmax



卷积层

- 卷积层如何检测特征

  - 检测复杂边缘

    将权重作为参数，在训练中学习。

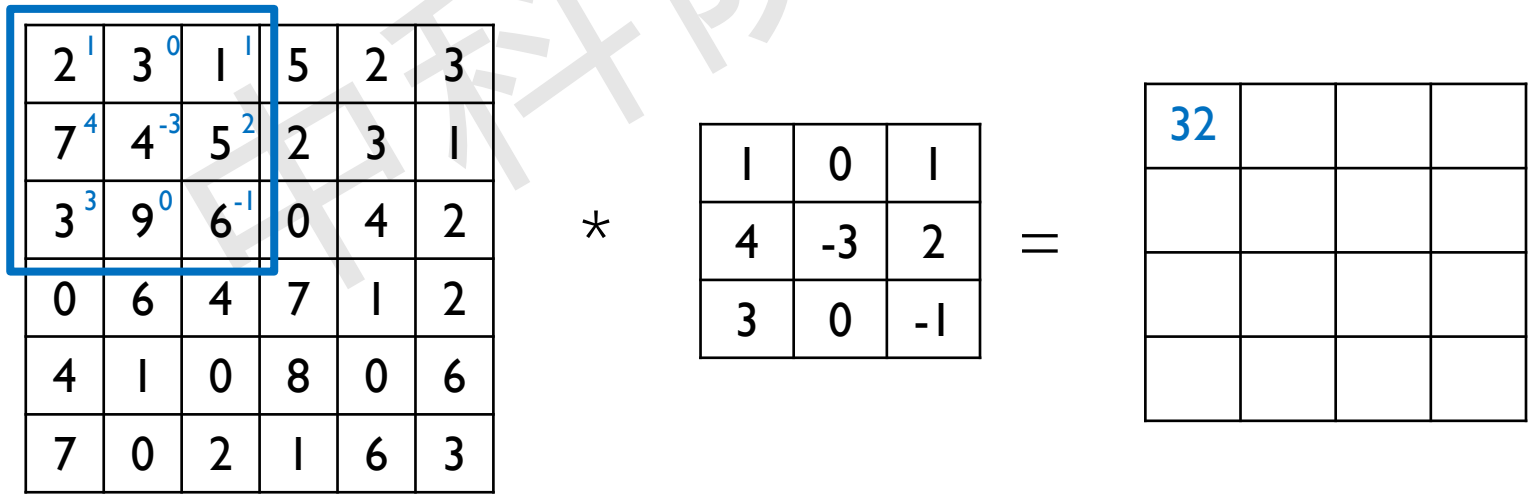
w0	w1	w2
w3	w4	w5
w6	w7	w8

filter/kernel

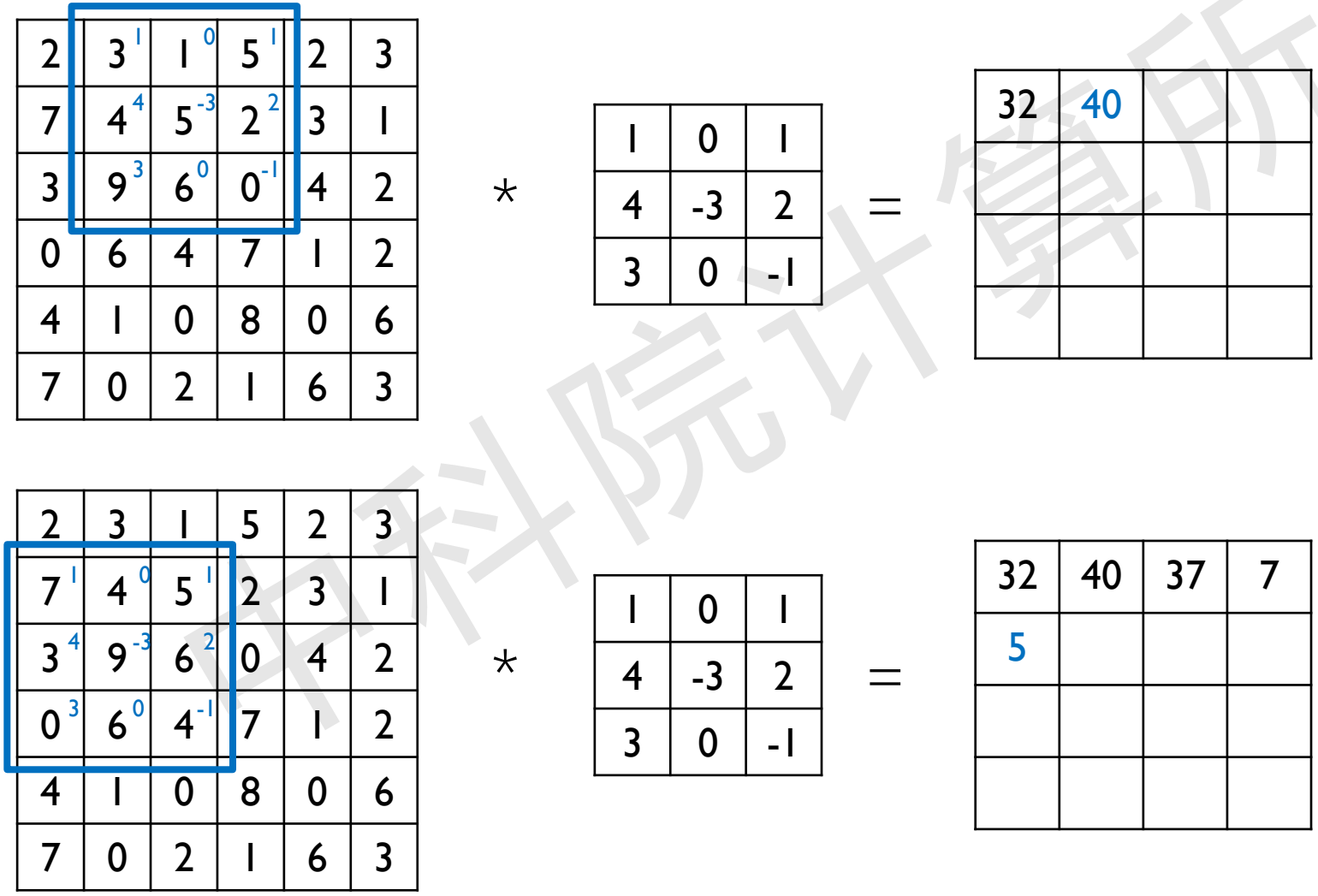
> ➢ 卷积神经网络的两个重要特征：**局部连接、权重共享**
>
> 可有效减少权重参数，避免过拟合，为增加卷积层数提供可能。

卷积层

- 卷积运算
  - 数学：$$y(n) = \sum_{i=-\infty}^{\infty} x(i)h(n-i) = x(n) * h(n)$$ （“*”表示卷积）
  - 神经网络：实际为计算矩阵内积(相关系数)；

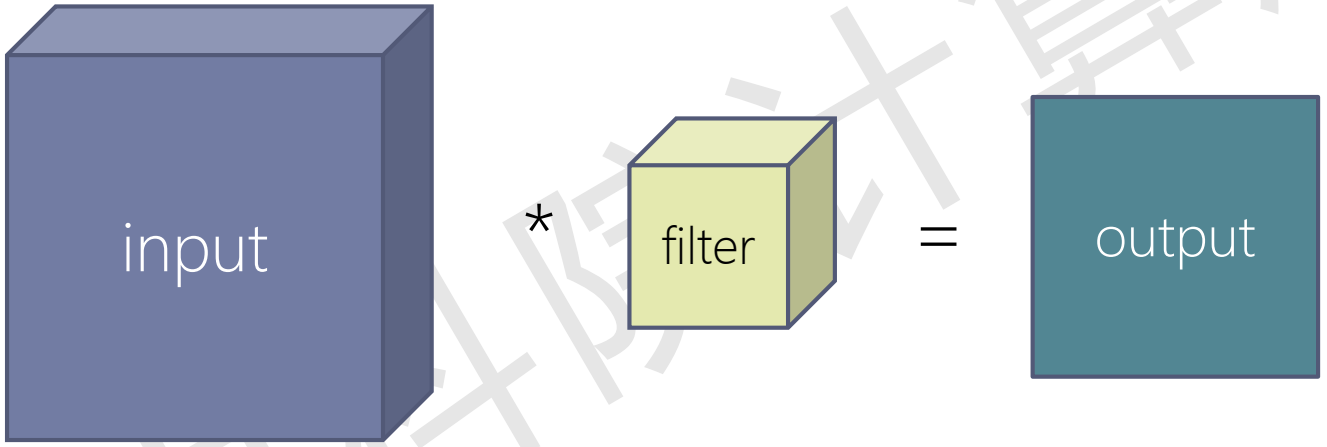


卷积层

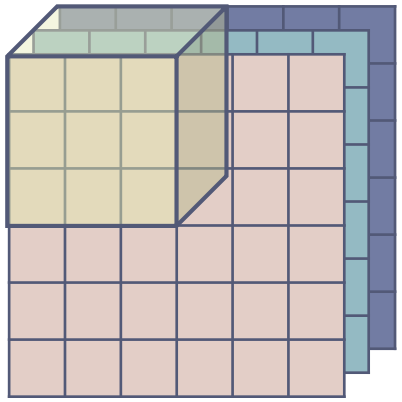

2 3 1 1 0 5 1 2 3
7 4 4 5 -3 2 2 3 1
3 9 3 6 0 0 -1 4 2
0 6 4 7 1 2
4 1 0 8 0 6
7 0 2 1 6 3
*
1 0 1
4 -3 2
3 0 -1
=
32 40
2 3 1 5 2 3
7 1 4 0 5 1 2 3 1
3 4 9 -3 6 2 0 4 2
0 3 6 0 4 -1 7 1 2
4 1 0 8 0 6
7 0 2 1 6 3
*
1 0 1
4 -3 2
3 0 -1
=
32 40 37 7
5

卷积层

- 多输入特征图单输出特征图卷积运算

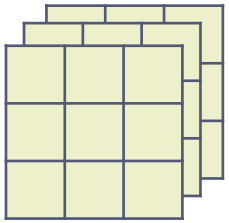


卷积层



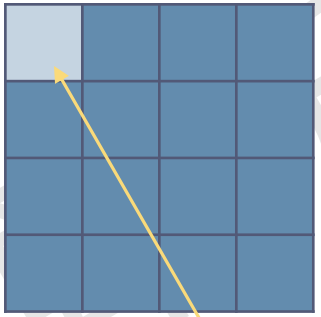
6x6x3

*



3x3x3

=



4x4

C = 0

0	0	0
0	2	2
0	1	2

-1	1	1
-1	1	-1
1	-1	1

C = 1

0	0	0
0	0	2
0	1	2

*

1	-1	-1
-1	0	-1
-1	0	1

=

2-2-1+2
+
0-2+0+2
+
(-1)+0+0+2

=

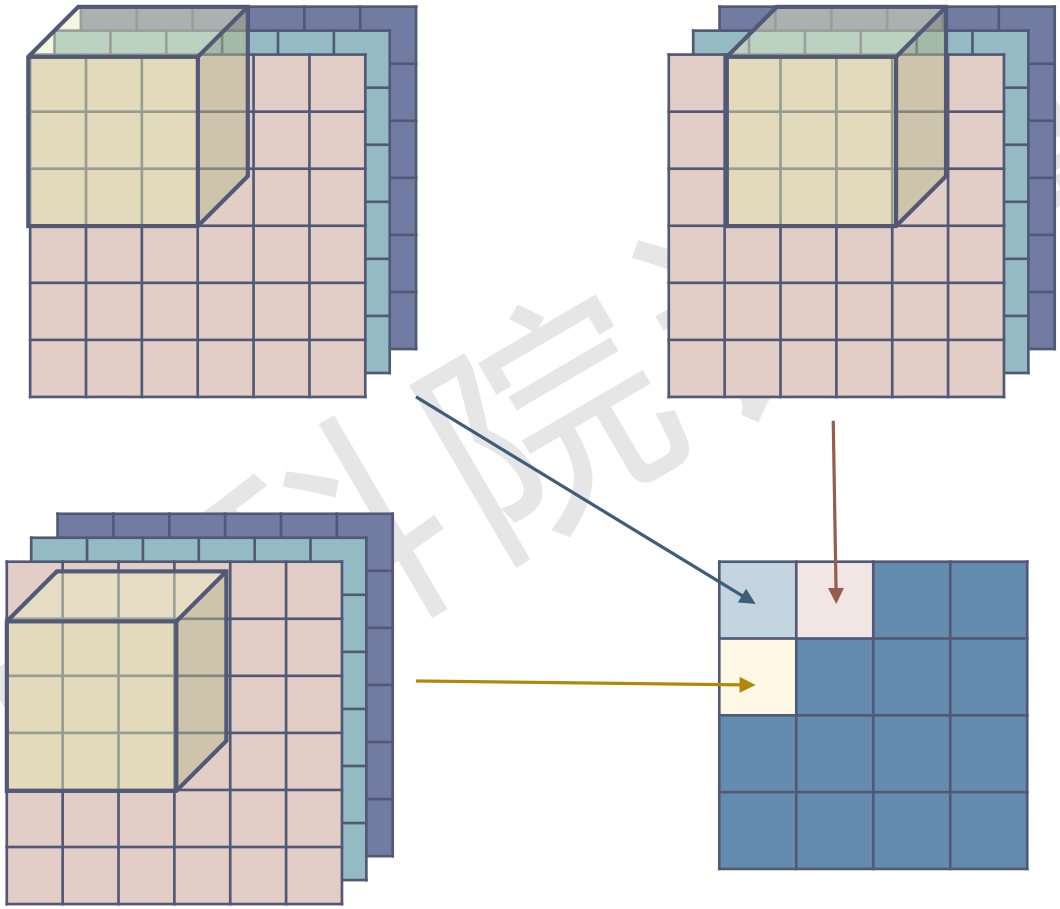
2

C = 2

0	0	0
0	1	1
0	0	2

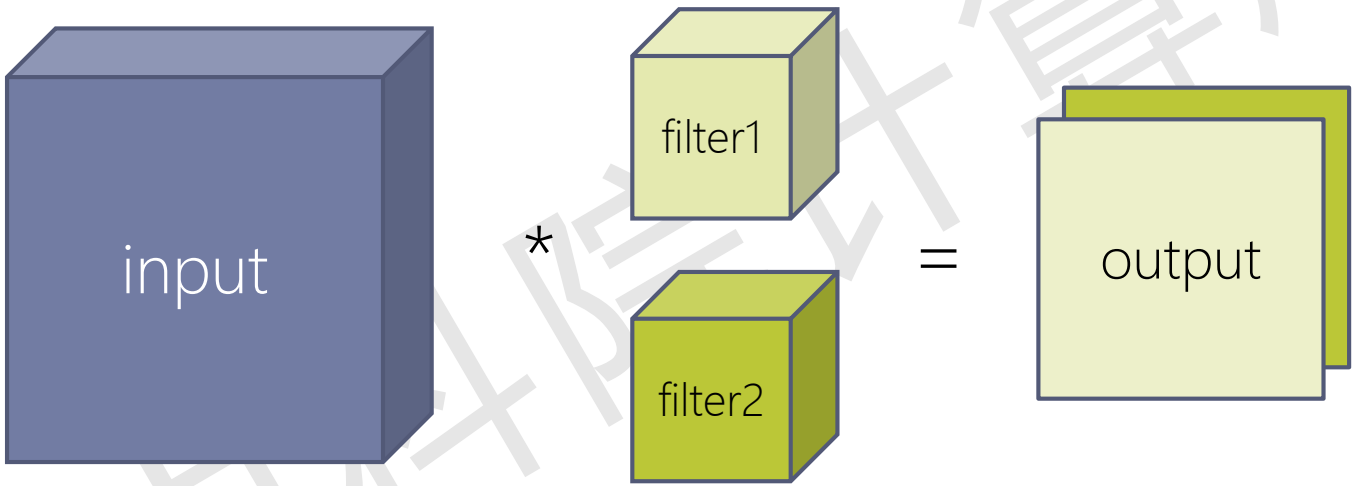
1	-1	-1
-1	-1	0
-1	1	1

卷积层



卷积层

- 多输入特征图多输出特征图卷积运算

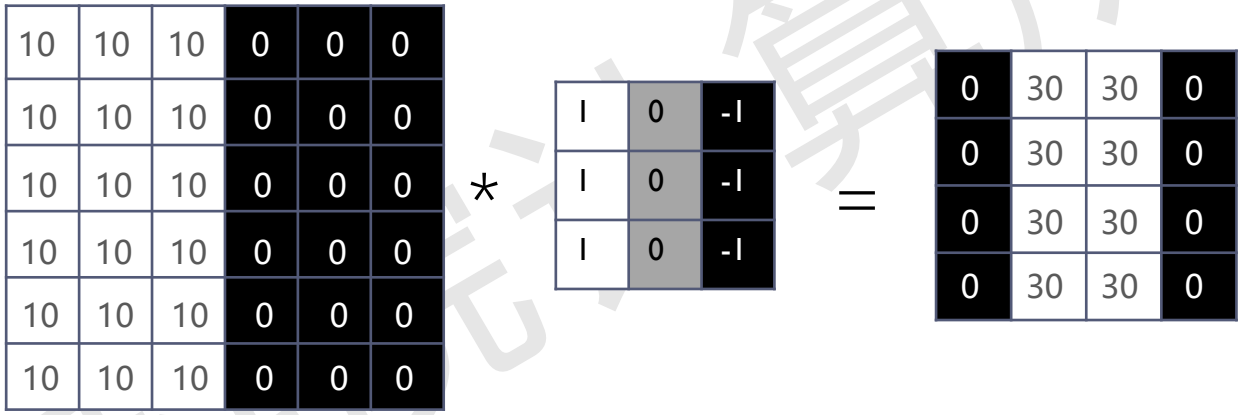


- 不同的过滤器可检测不同特征。

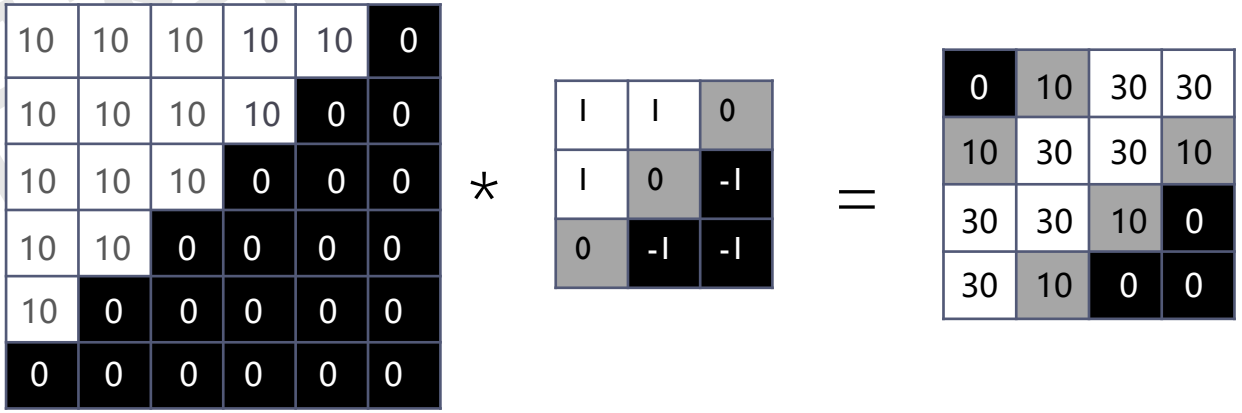
卷积层

- 卷积层如何检测特征

- 检测垂直边缘

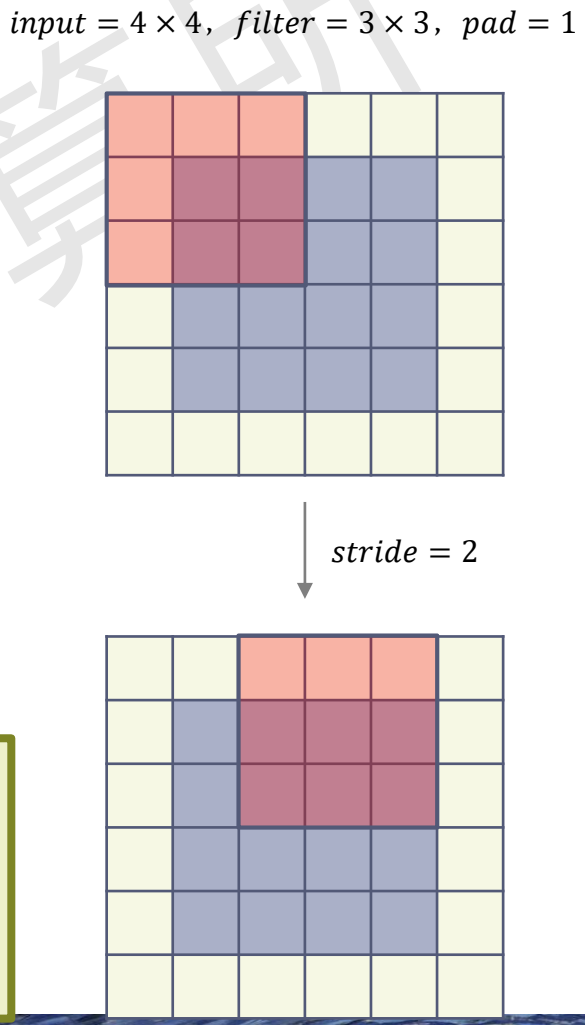


- 检测对角线边缘



卷积层

- 边界扩充(padding)
  - 扩大输入图像/特征图的尺寸并填充像素
  - 防止深度网络中图像被动持续减小
  - 强化图像边缘信息
- 卷积步长(stride)
  - 滑动滤波器时每次移动的像素点个数
  - 与pad共同确定输出图像尺寸

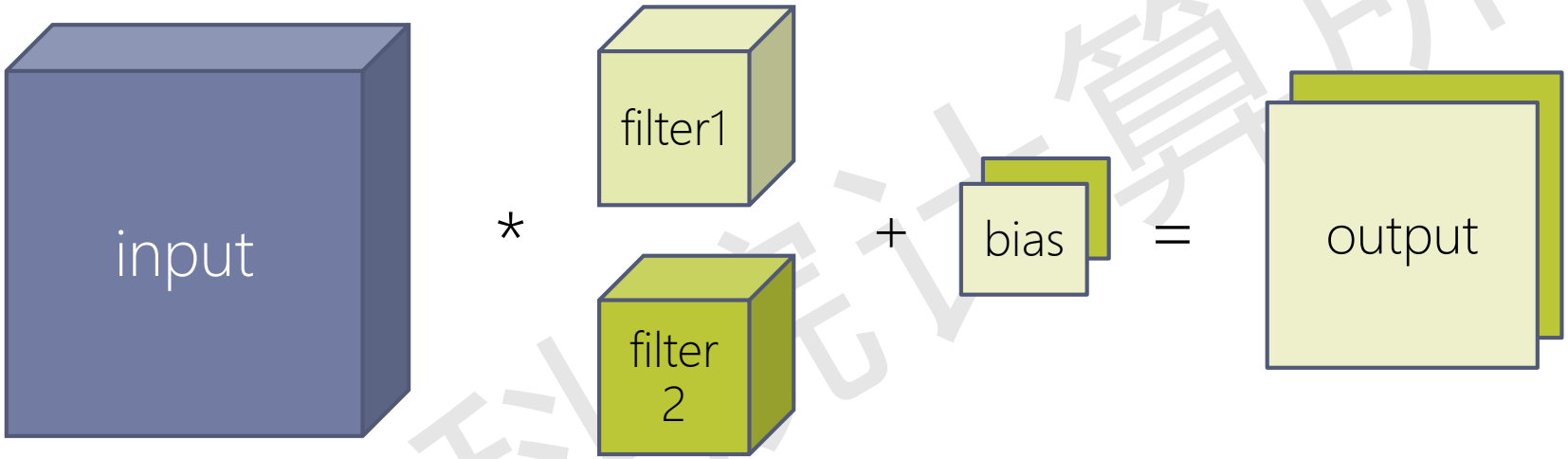


$$input = w_i \times h_i,\ filter = f \times f,\ pad = p,\ stride = s$$

$$output = \left\lfloor \frac{w_i + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h_i + 2p - f}{s} + 1 \right\rfloor$$

卷积层

- 总结：卷积层参数



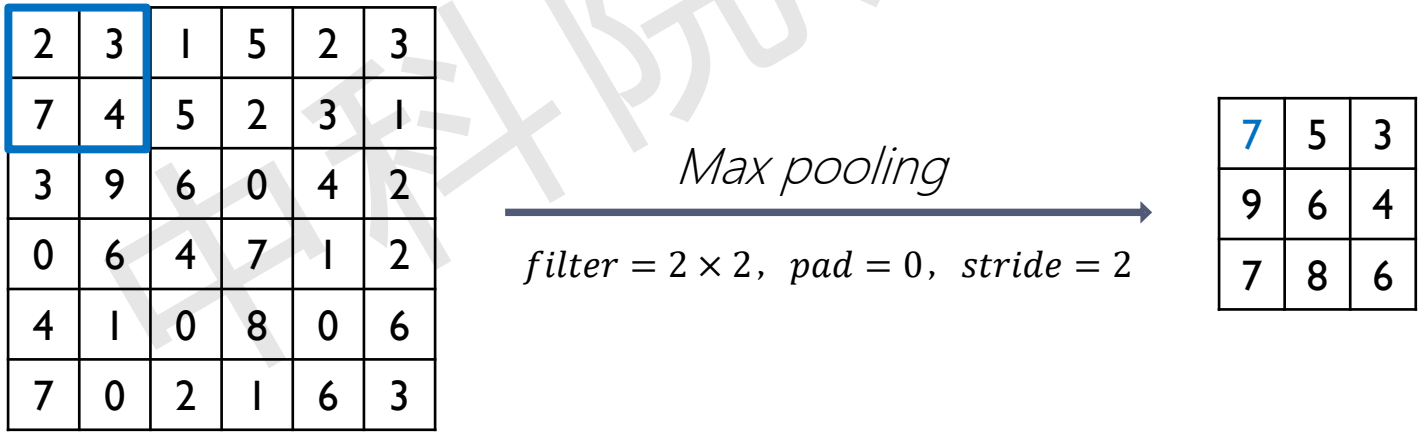
$w_i \times h_i \times c_i$　　$c_o \times f \times f \times c_i$　　$1 \times 1 \times c_o$　　$w_o \times h_o \times c_o$

- stride
- pad
- $\text{output} = \left[\frac{w_i+2p-f}{s}+1\right] \times \left[\frac{h_i+2p-f}{s}+1\right]$

- filter：可训练
- bias：可训练，使分类器偏离激活函数原点，更灵活；
- activation

池化层

- Pooling
  - Max Pooling / Avg Pooling / L2 Pooling
  - 主动减小图片尺寸，从而减少参数的数量和计算量，控制过拟合；
  - 不引入额外参数；



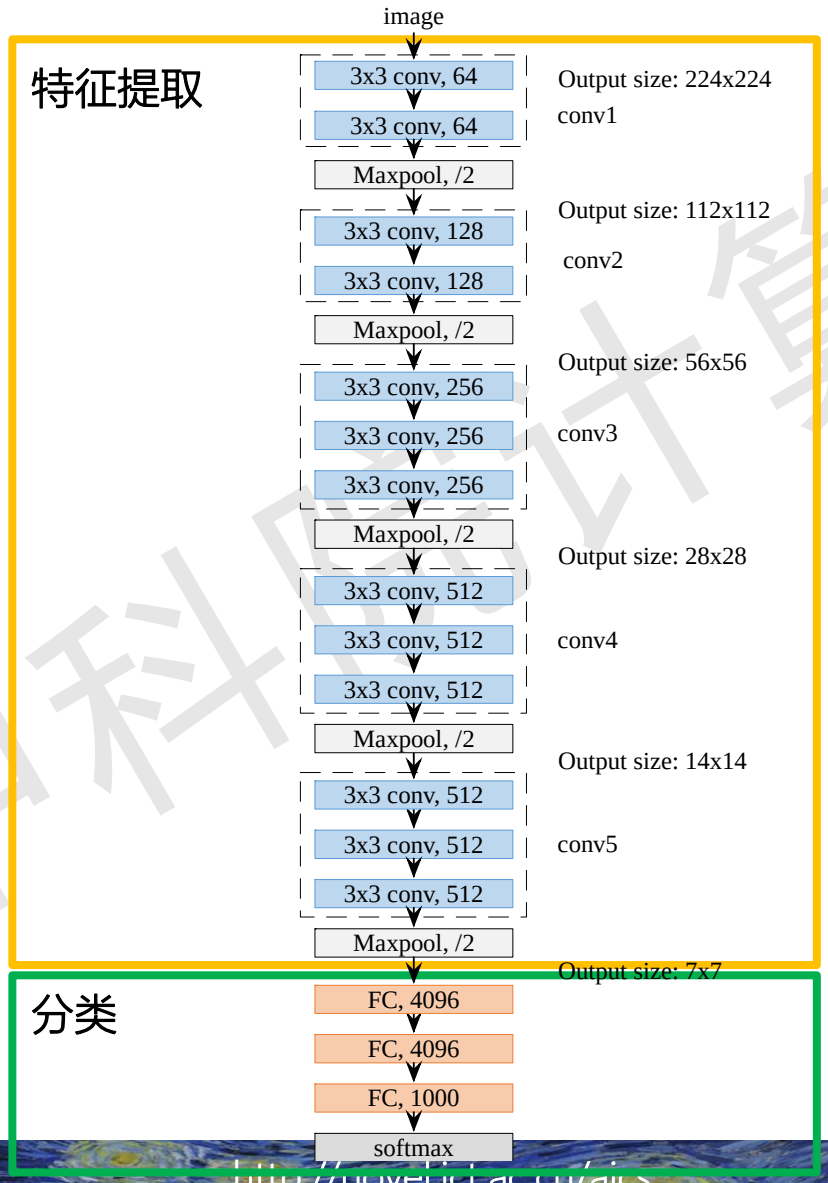
Max pooling 可保留特征最大值，提高提取特征的鲁棒性。

全连接层

- Fully Connect
  - 卷积层和池化层构成特征提取器，全连接层则为分类器；
  - 将特征提取得到的高维特征图映射成一维特征向量，该特征向量包含所有特征信息，可转化为各个类别的概率。
- Softmax
  - 通常作为网络的最后一层，对输出进行归一化，输出分类概率；
  - 凸显其中最大的值并抑制远低于最大值的其他分量；
  - Softmax层输入、输出数据规模相同；
  - 公式：$f(z_j) = e^{z_j} / \sum_{i=0}^{n} e^{z_i}$

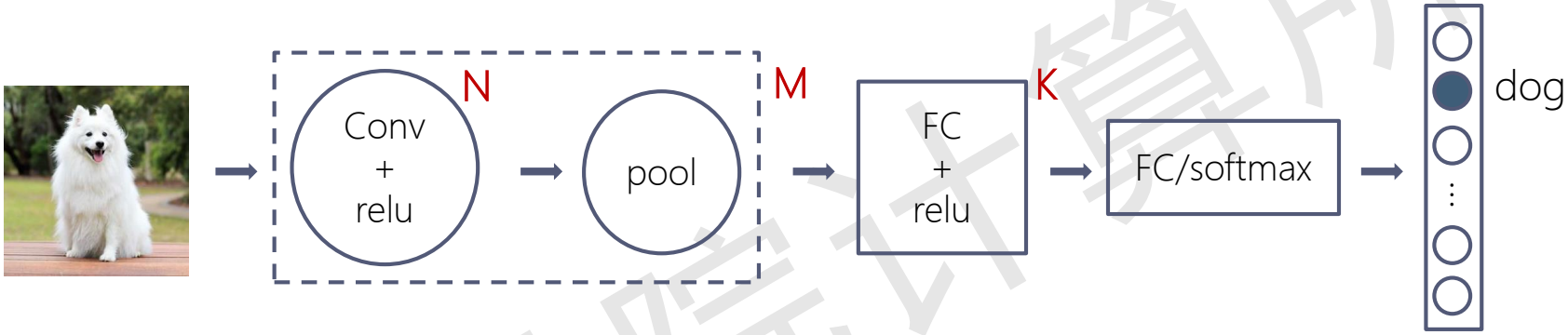
各层如何排布组成一个网络?

- VGG16



卷积神经网络结构

- 层排列规律



- 常见卷积神经网络由卷积层（激活）、池化层和全连接层构成；
- 各层的常见排列方式如图所示，其中N、M、K为重复次数；
- 例如：N=3，M=1，K=2 情况下的网络结构为：

$$\text{input} \rightarrow \text{conv(relu)} \rightarrow \text{conv(relu)} \rightarrow \text{conv(relu)} \rightarrow \text{pool} \rightarrow \text{FC(relu)} \rightarrow \text{FC(relu)} \rightarrow \text{FC} \rightarrow \text{output}$$

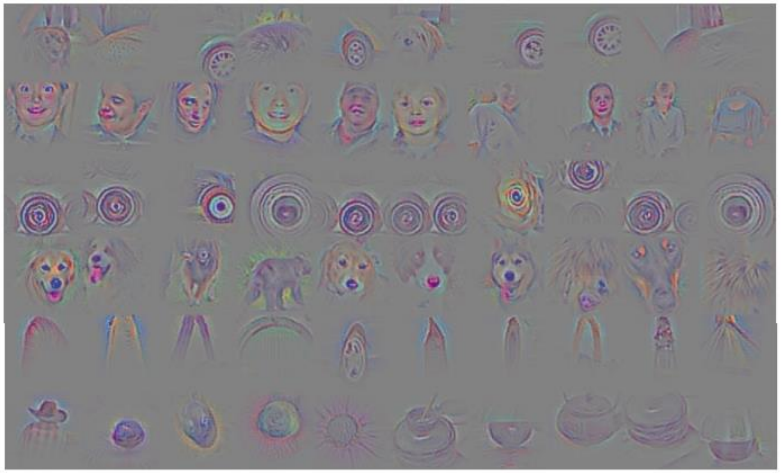
- 其中卷积和池化部分可包含分支和连接结构，将在具体网络分析中介绍。

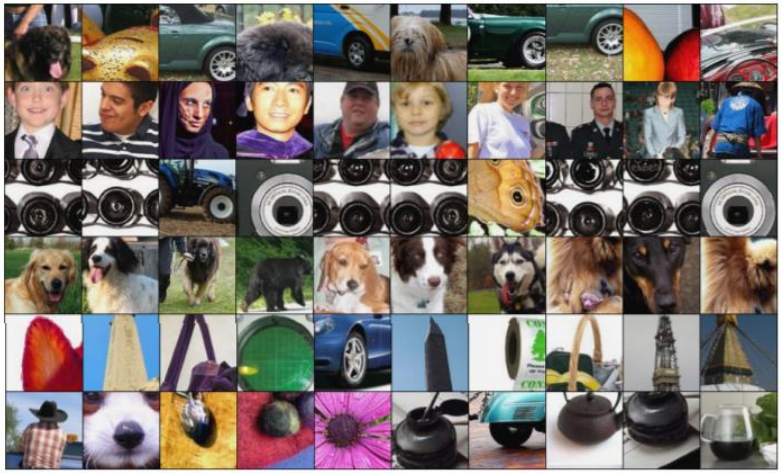
浅层学习局部特征，深层学习整体特征

- 神经网络可视化：

conv6

conv9





Springenberg, J. T.; Dosovitskiy, A.; Brox, T. & Riedmiller, M. Striving for simplicity: the all convolutinal net ICML, 2015, 1-12

卷积神经网络结构

- 为何选择“深”而非“广”的网络结构

➢ 即使只有一层隐层，只要有足够的神经元，神经网络理论上可以拟合任意连续函数。为什么还要使用深层网络结构？

- 深度网络可从局部到整体“理解图像”

学习复杂特征时（例如人脸识别），浅层的卷积层感受野小，学习到局部特征，深层的卷积层感受野大，学习到整体特征。

- 深度网络可减少权重数量

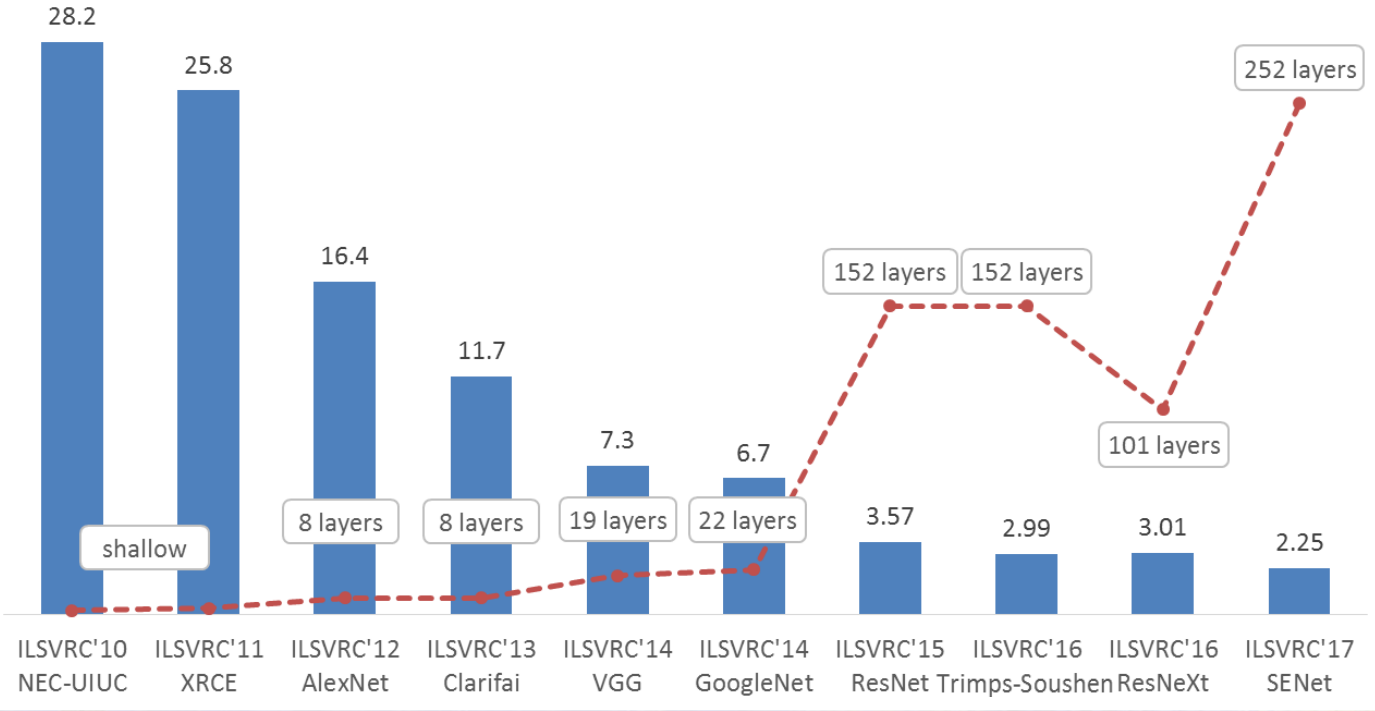
以宽度换深度，用多个小卷积替代一个大卷积，在获得更多样特征的同时所需权重数量也更少。

提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

基于CNN的图像分类算法

- 对卷积神经网络的研究可追溯至日本学者福岛邦彦提出的neocognition模型。在其1979 和1980年发表的论文中，福岛参照生物的视觉皮层（visual cortex）设计了以“neocognition”命名的神经网络。
- AlexNet使用卷积神经网络解决图像分类问题，在ILSVRC2012中取得获胜并大大提高了state-of-art的准确率。自此卷积神经网络在图像分类领域获得快速发展。

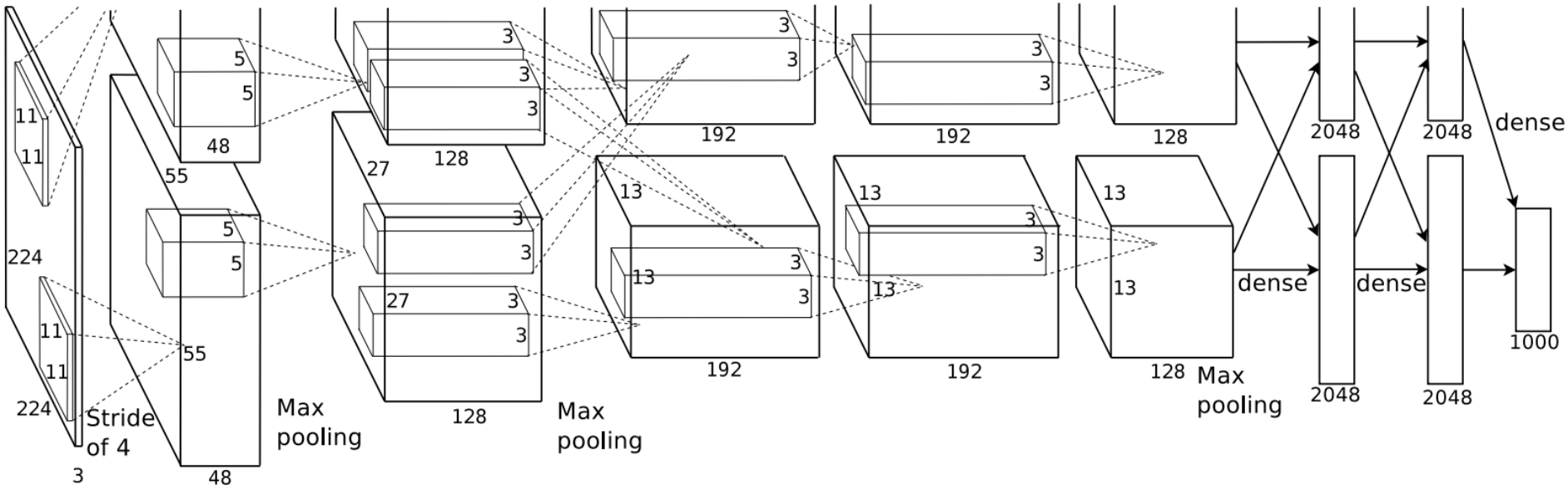


基于CNN的图像分类算法

- AlexNet
- VGG
- Inception 系列
- ResNet

AlexNet

- Paper: ImageNet Classification with Deep Convolutional Neural Networks (2012)
- Author: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton
- Test: error rate on ImageNet, top1: 37.5%, top5: 17.0%



论文中给出的网络结构（用两台GPU训练）

AlexNet

层号	层名称	输入大小	卷积核或池化窗口大小	输入通道数	输出通道数	步长
1	Conv	224 × 224	11 × 11	3	96	4
	ReLU	55 × 55	–	96	96	1
	LRN	55 × 55	–	96	96	1
	MaxPool	55 × 55	3 × 3	96	96	2
2	Conv	27 × 27	5 × 5	96	256	1
	ReLU	27 × 27	–	256	256	1
	LRN	27 × 27	–	256	256	1
	MaxPool	27 × 27	3 × 3	256	256	2
3	Conv	13 × 13	3 × 3	256	384	1
	ReLU	13 × 13	–	384	384	1
4	Conv	13 × 13	3 × 3	384	384	1
	ReLU	13 × 13	–	384	384	1
5	Conv	13 × 13	3 × 3	384	256	1
	ReLU	13 × 13	–	256	256	1
	MaxPool	13 × 13	3 × 3	256	256	2
6	FC	–	–	9216	4096	1
	ReLU	–	–	4096	4096	1
	Dropout					
7	FC	–	–	4096	4096	1
	ReLU	–	–	4096	4096	1
	Dropout					
8	FC	–	–	4096	1000	1
	softmax	–	–	1000	1000	1

What's New?

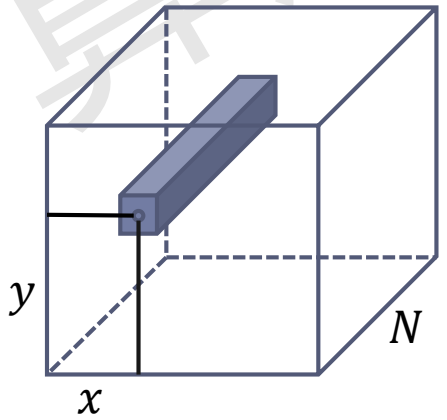
- ReLU激活函数：训练中收敛速度更快；
- LRN局部归一化：提升较大响应，抑制较小响应；
- MaxPool：避免特征被平均池化模糊，提升特征鲁棒性；
- Dropout：随机舍弃部分隐层节点，避免过拟合；

AlexNet

- Local Respose Normalization (LRN)

- 局部响应归一化；

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$



- 对图像的每个“位置”，提升高响应特征，抑制低响应特征；
- 减少高激活神经元数量，提高训练速度，抑制过拟合；

被后来研究者发现无明显效果，故现在很少使用。

AlexNet

- Dropout

- 随机丢弃部分神经元
- 在模型训练过程中，以一定概率随机地舍弃某些隐层神经元。在反向传播更新权重时，不更新与该神经元相关的权重；
- 但是与被舍弃神经元相关的权重得保留下来（只是暂时不更新），另一批样本输入时继续使用与该神经元相关的权重；
- 防止训练数据中复杂的co-adaptation，抑制过拟合；

AlexNet

- AlexNet 成功的原因

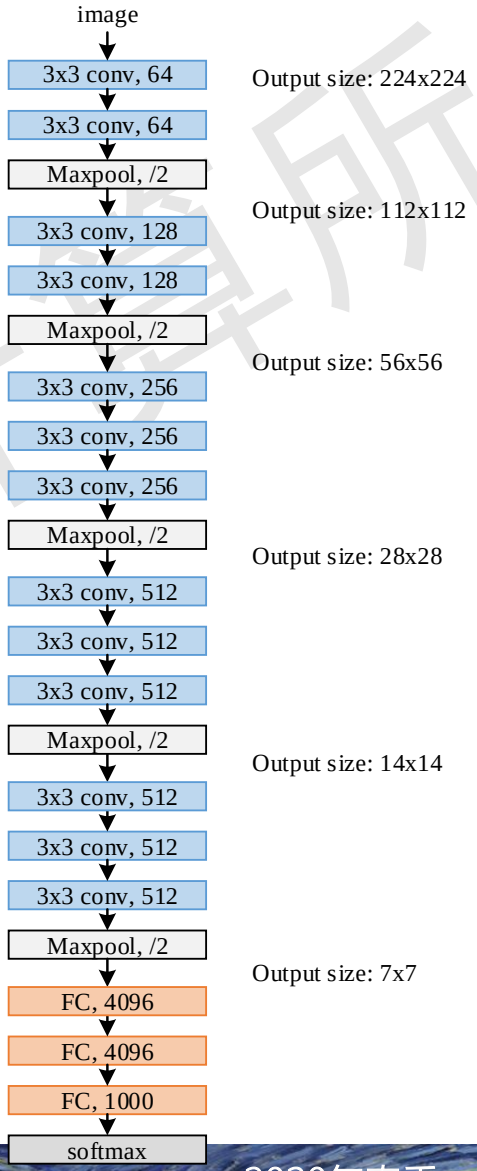
- 使用多个卷积层，有效提取图像特征；
- ReLU帮助提高训练速度；
- Dropout、数据增强扩大训练集，防止过拟合。

使用更多卷积层是否能进一步提升效果？

VGG

- Paper: Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)
- Author: K. Simonyan, A. Zisserman
- Test: error rate on ImageNet, top1: 24.7%, top5: 7.5%

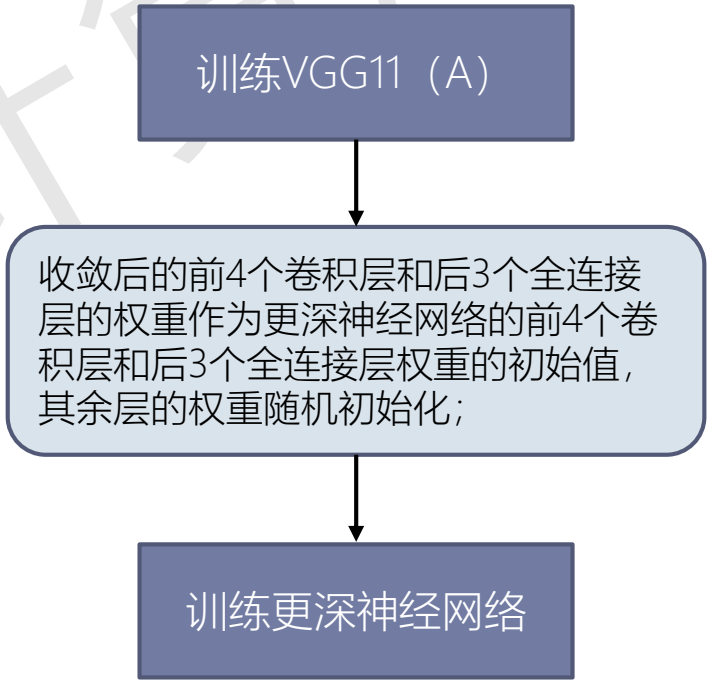
- VGG16



VGG

- 由简单到复杂的网络结构

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



VGG

- 实验结果

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

- A/A-LRN：加 LRN 准确率无明显提升；
- A/B/D/E：层数越多准确率越高；
- C/D： conv3x3 比 conv1x1 得到的准确率高；
- Multi-scale training 可明显提高准确率；

VGG

- 规整的卷积-池化结构

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input				224 × 224 × 3	
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool				112 × 112 × 64	
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool				56 × 56 × 128	
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool				28 × 28 × 256	
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool				14 × 14 × 512	
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool				7 × 7 × 512	

- Conv
  - 所有卷积层filter大小/stride/pad相同；
  - filter=3*3，stride=1，pad=SAME；
    - pad=SAME：补pad至输出图像大小等于输入图像大小
    - pad=VALID：pad=0；
- Maxpool
  - 所有池化层filter大小/stride/pad相同；
  - filter=2*2，stride=2，pad=0；

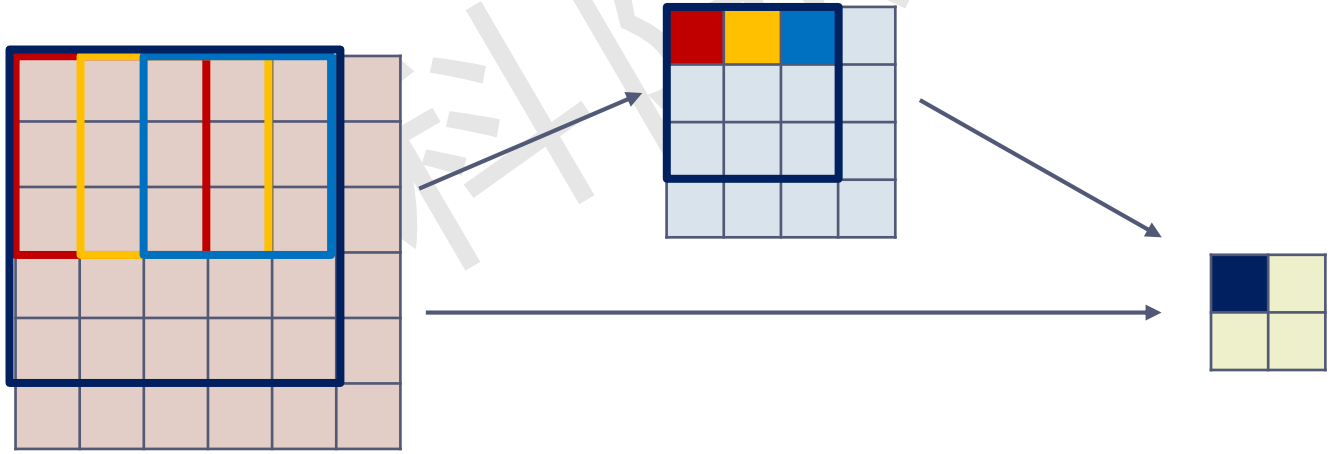
卷积层：负责数据体深度变换（控制特征图数量）

池化层：负责数据体长宽变换（控制特征图大小）

VGG

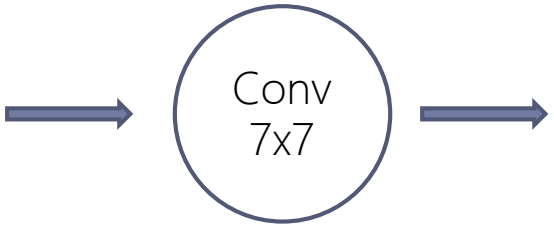
- 规整的<卷积-池化>结构

- 多层小卷积比单层大卷积效果好;
- 实验：对 VGG13(上表B)，使用5x5conv代替两层3x3conv，进行训练和测试;
- 原因：一个5x5conv和两个3x3conv的感受野大小相同；每个卷积层加入ReLU，两层3x3conv 决策函数的区分能力更强



- 结果： 5x5conv网络比两个3x3conv网络top-1准确率低7%。

相同感受野，多层网络权值更少



$$weight\_count_1 = 7 \times 7 = 49$$

$$weight\_count_2 = 3 \times 3 \times 3 = 27$$

VGG

- VGG成功的原因

- 更深的卷积神经网络，更多的卷积层和非线性激活函数，提升分类准确率
- 使用规则的多层小卷积替代大卷积，减少参数数量，提高训练收敛速度
- 部分网络层参数的预初始化，提高训练收敛速度

卷积核还能不能更小？网络还能不能更深？

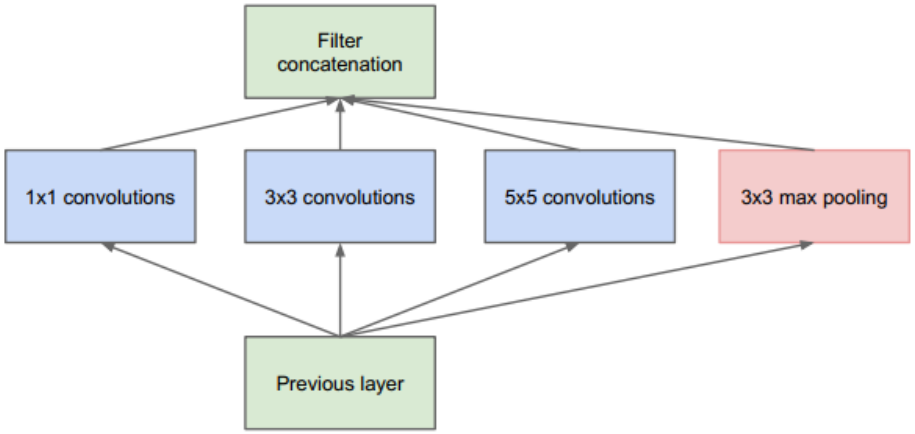
Inception

- Inception-v1(GoogLeNet)：Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions, CVPR 2015:1-9.
- BN-Inception：Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. ICML, 2015:448-456.
- Inception-v2,Inception-v3：Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[C]// CVPR, 2016:2818-2826.
- Inception-v4：Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, AAAI'2017.

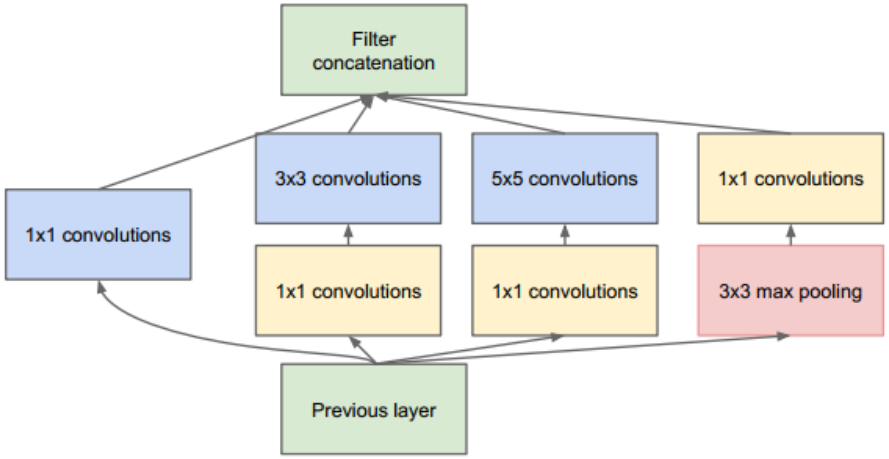
网络	主要创新	Top5错误率	网络层数
GoogLeNet	提出inception结构	6.67%	22
BN-Inception	提出 Batch Normalization, 用3x3代替5x5	4.82%	—
Inception-v3	将一个二维卷积拆成两个一维卷积，辅助分类器的全连接层做BN	3.5%	42
Inception-v4	inception模块化，结合ResNet的跳转结构	3.08%	—

Inception-v1

- Inception 模块



(a) Inception module, naïve version



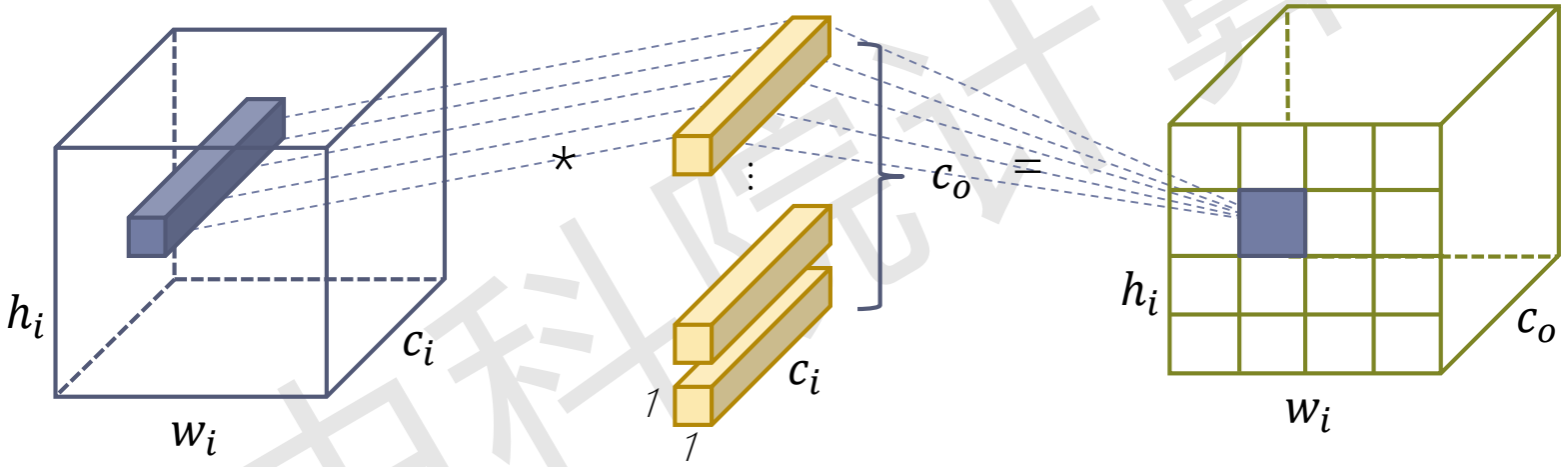
(b) Inception module with dimension reductions

- Naïve version：叠加多种尺寸的卷积层和池化层，获得不同尺度的特征，提高网络对不同尺寸特征的适应性；
- Dimension redutions：使用1x1的卷积层来缩减维度(减小channel)，形成"瓶颈层"，减少参数。

Inception-v1

- 1x1卷积

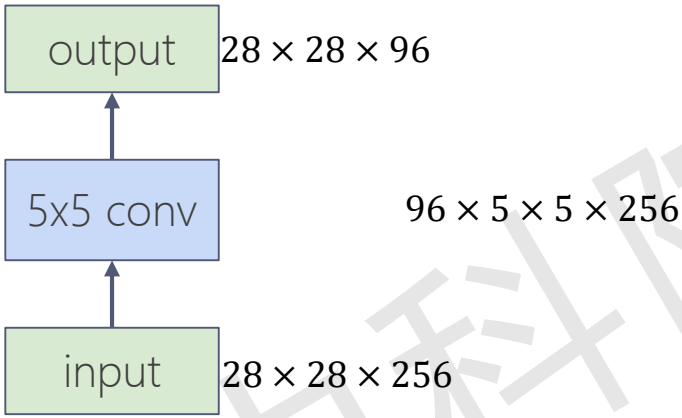
- 作用：跨通道聚合，进一步可以起到降维（或者升维）的作用，减少参数。



- 相当于在输入和输出之间做了一个特征上的全连接，提取得到非线性特征；
- 同时，当 co < ci 时，维度降低，参数减少；
- *扩展：Network in Network*

Inception-v1

- 1x1卷积
- 使用1x1卷积，形成"瓶颈层"，可有效减少计算量和参数数量；

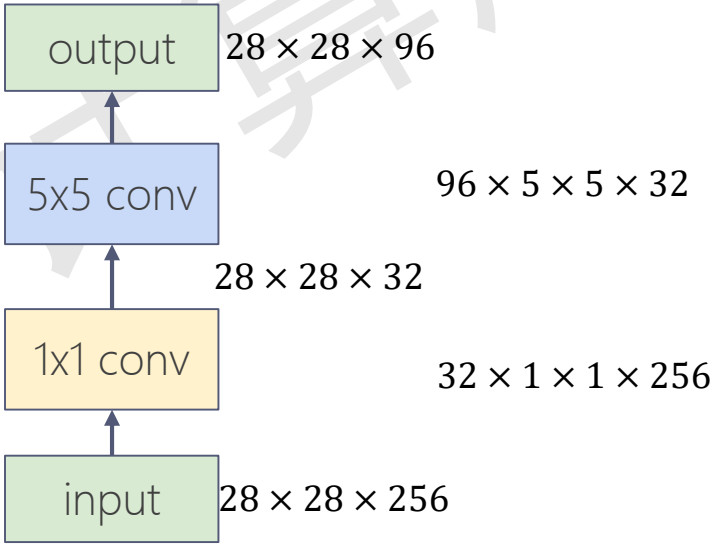


乘法次数：

28×28×96×5×5×256≈4.8×10^8

参数数量：

96×5×5×256 ≈6.1×10^5



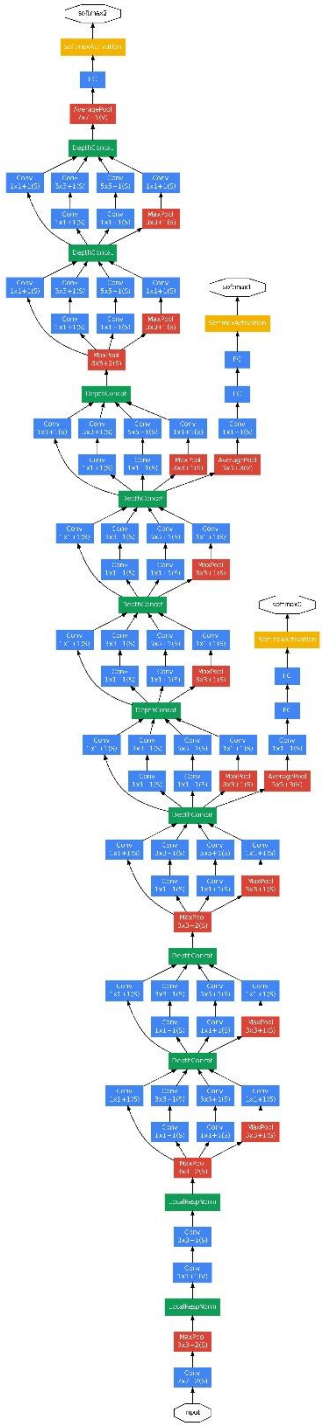
乘法次数：

28×28×32×256+28×28×96×5×5×32≈6.7×10^7

参数数量：

32×256+96×5×5×32 ≈8.5×10^4

Inception-v1



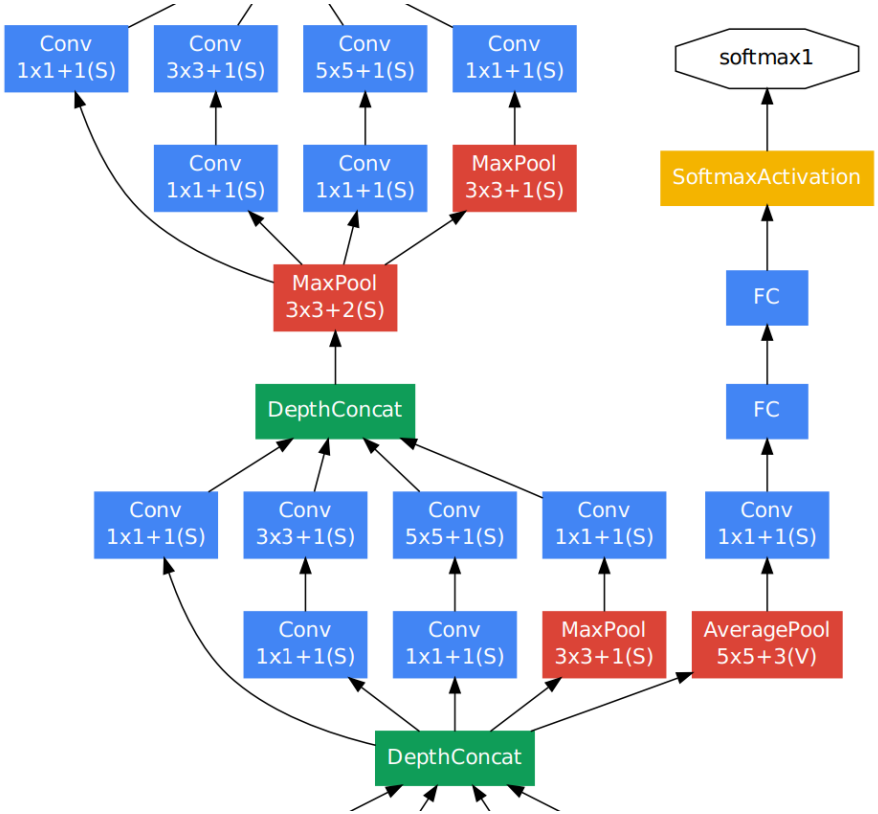
- GoogLeNet 网络结构

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

Inception-v1

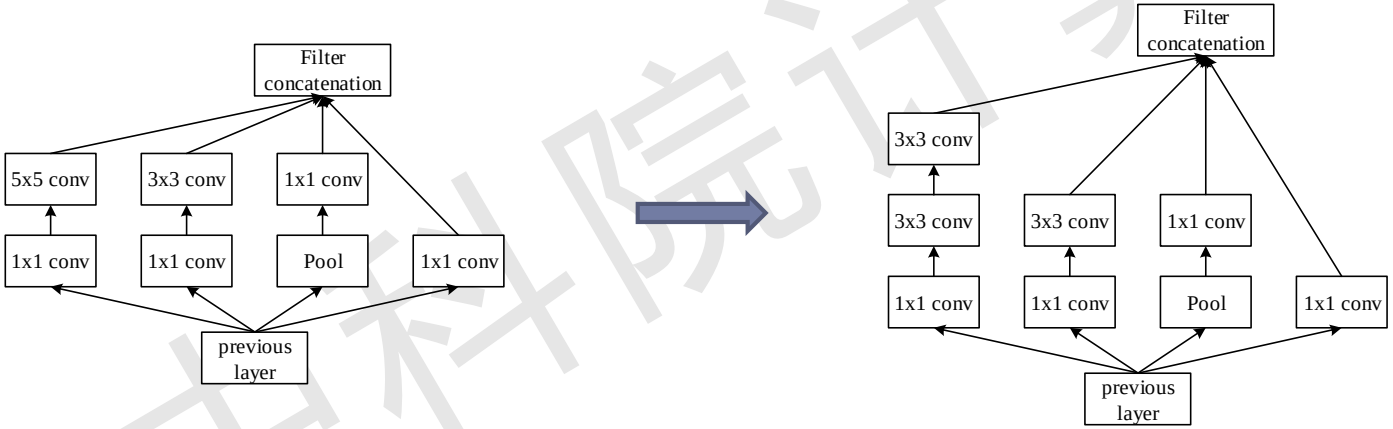
- Softmax 辅助分类网络



- 训练时，让中间某一层的输出经过softmax得到分类结果，并按较小的权重加到最终分类结果中，相当于模型融合。防止多层神经网络训练过程中梯度消失。
- 推断时，softmax辅助分类网络会被去掉。

BN-Inception

- 学习 VGG 用两个 3x3 卷积代替一个 5x5 卷积
  - 具体分析见 VGG 部分，不再赘述。



- 使用 BatchNorm，并在每个卷积层之后、激活函数之前插入 BN 层

BN-Inception

- BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- normalize
  - 将激活层的输入调整为标准正态分布（均值为0，方差为1）；
  - 激活层输入分布在激活函数敏感部分，输入有小变化就能是损失函数有较大的反应，避免梯度消失，加快训练速度。
- Scale and shift
  - 标准化后的输入使得网络的表达能力下降；
  - 为保持网络的表达能力，增加两个可训练参数。

BN-Inception

- BatchNorm 效果

- BN 可替代 LRN / Dropout / L2 Normalization；
- 可提高收敛速度、训练速度；
- 可选择更高的学习率，方便调参；

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

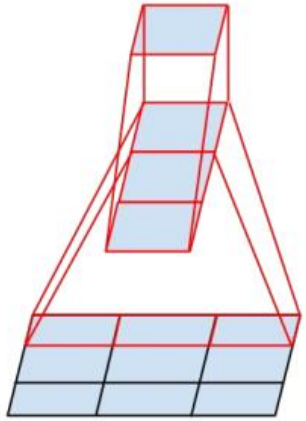
Figure 3. For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.

- -x5 表示学习率设为inception初始学习率的5倍。

Inception-v3

- Factorization 思想

- 将 3x3 卷积拆分成 1x3 和 3x1 卷积；
- 减少参数数量，同时通过非对称的卷积结构拆分增加特征多样性；



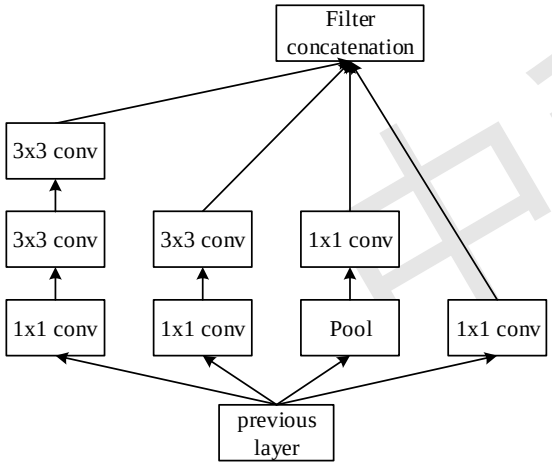


Figure 5

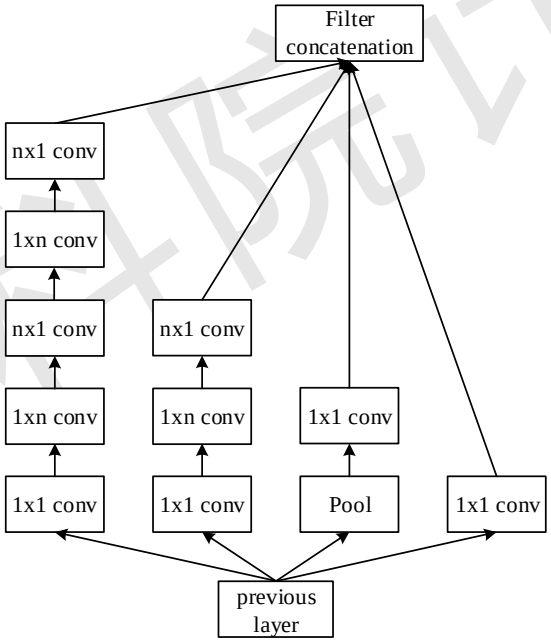


Figure 6

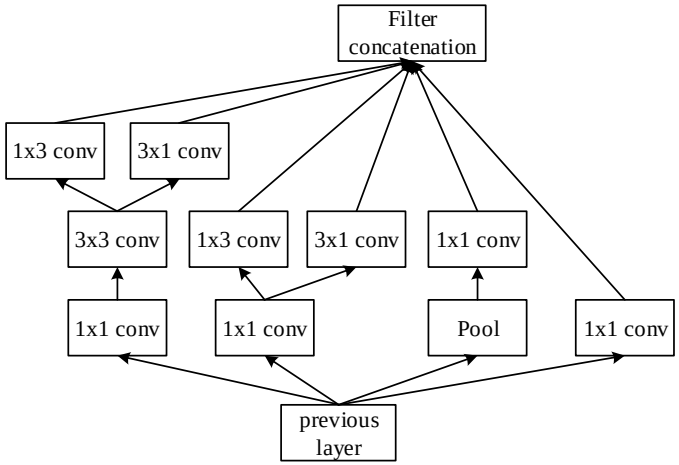


Figure 7

Inception-v3

- 网络结构

- 将前面三种 inception 结构组合起来
- GoogLeNet中7x7卷积拆分成3x3卷积
- 卷积层和辅助分类器的全连接层做BN

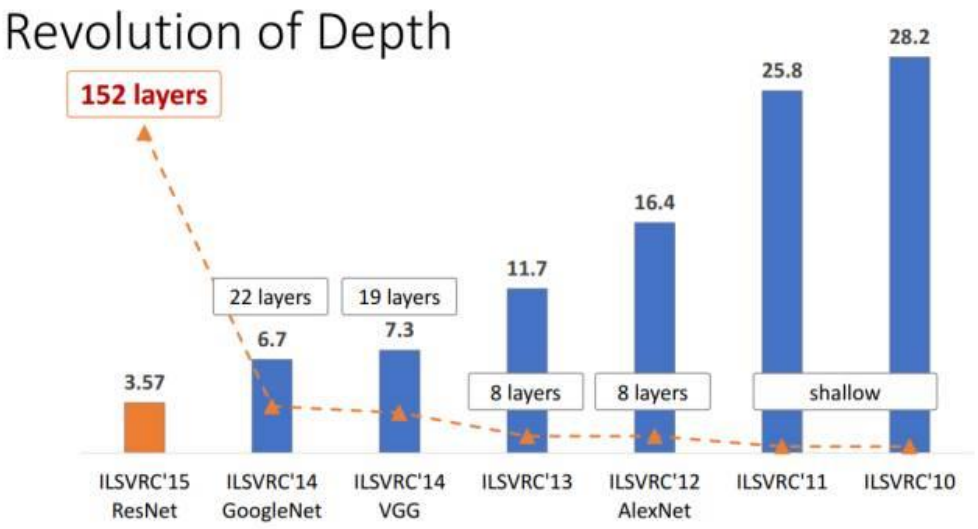
type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

ResNet

- Paper: Deep Residual Learning for Image Recognition (2015)
- Author: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
- Test: error rate on ImageNet, top5: 3. 57% (resnet152)

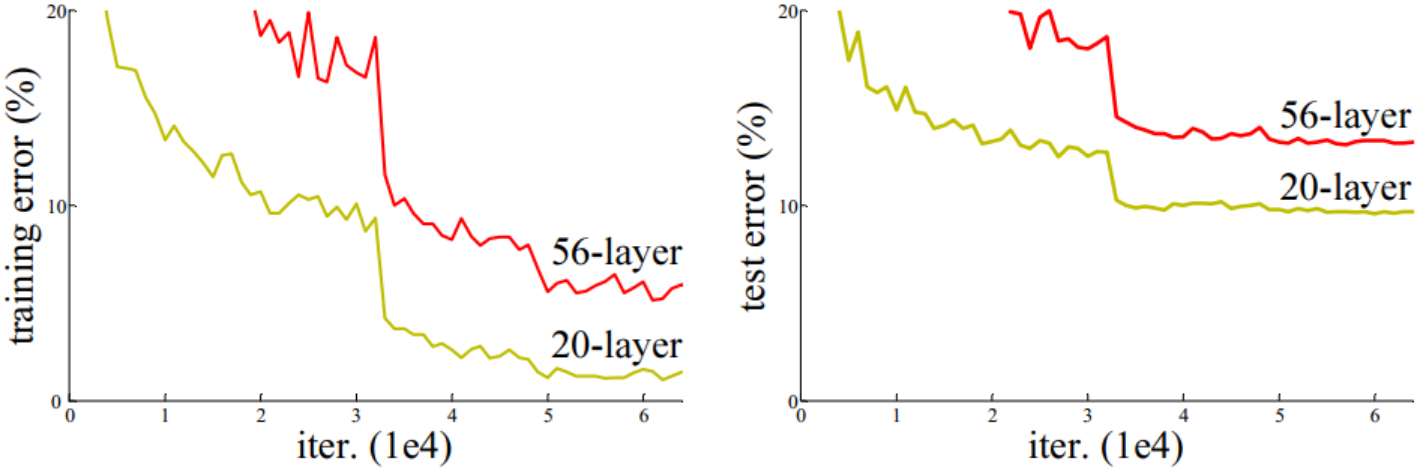
ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd



ResNet

- 问题：卷积层堆积就能提升图像分类准确率吗？
- 实验：分别用20层和56层卷积神经网络在cifar10数据集上进行训练和测试，发现更深的网络错误率更高，在ImageNet数据集上也同样如此。



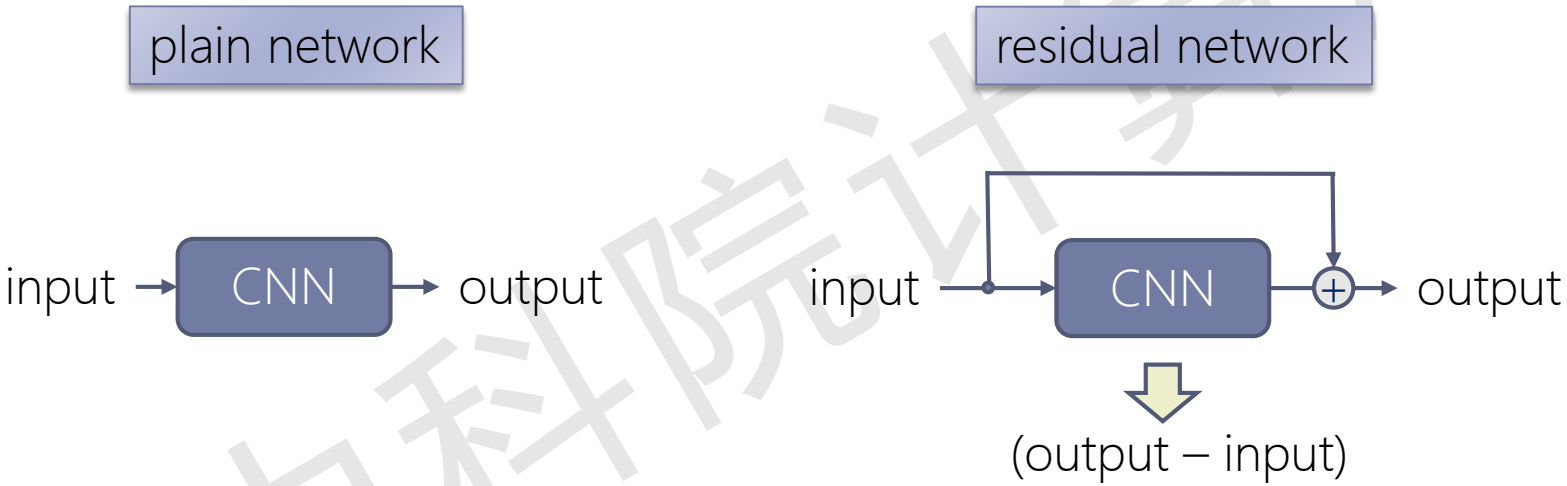
- 原因：梯度消失？ No, 使用BatchNorm可有效缓解梯度消失；

  过拟合？ No, 更深的网络在训练集上的误差同样更高；

  神经网络退化：收敛到极值点而非最值，误差大。

ResNet

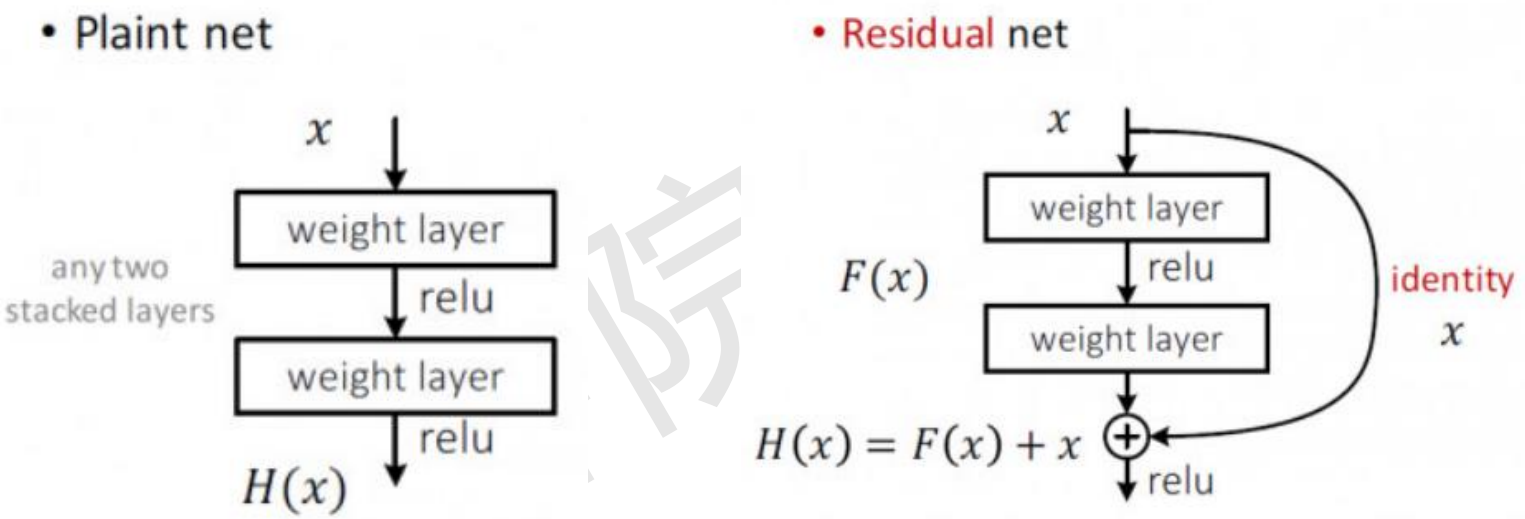
- 什么是"残差"



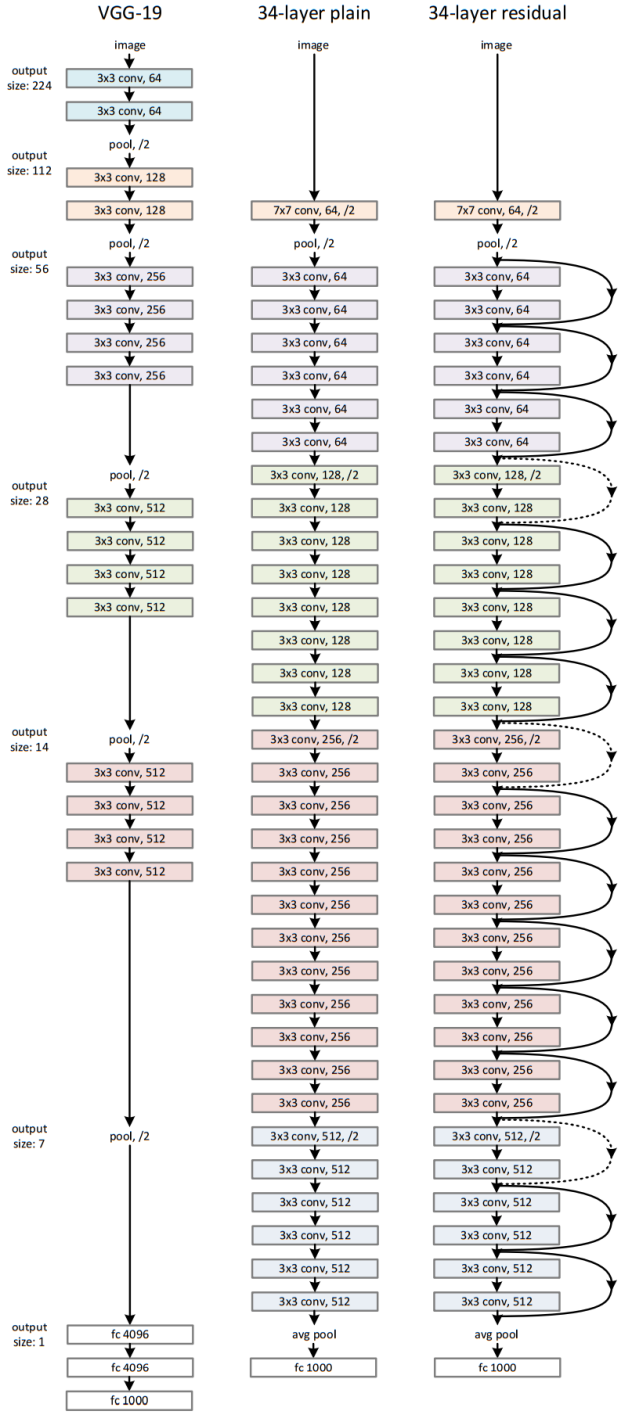
- Plain network(普通网络)：直接用多项式拟合输出；
- Residual network(残差网络)：建立在BN之上，用多项式拟合差值；
- 优点：在解附近时权重的反应更灵敏，更容易学习获得最优解。

ResNet

- 残差块



- 残差网络在训练时更容易收敛



将残差块应用到普通网络

改造VGG得到 plain-network

- plain-network：无跳转连接的普通网络；
- 基本全部由卷积层构成：filter=3*3, stride=1, pad=SAME；
- 特征图尺寸的减小由 stride=2 的卷积层完成；
- 若特征图的尺寸不变，则特征的数量也不变；若特征图的尺寸减半，则特征图的数量翻倍；

增加跳转连接得到 resnet

- 实线：特征图尺寸和特征数量不变，直接相连；
- 虚线：特征图尺寸减半，特征图数量翻倍；
- 特征图数量翻倍的两种方法：

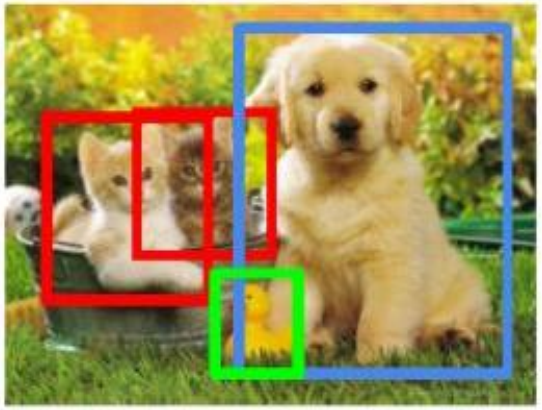
  a. 以stride=2直接取值，不够的特征补0(不引入额外参数)

  b. 用stride=2，特征数量翻倍的的1x1卷积做映射，卷积的权重经过学习得到，会引入额外参数；

提纲

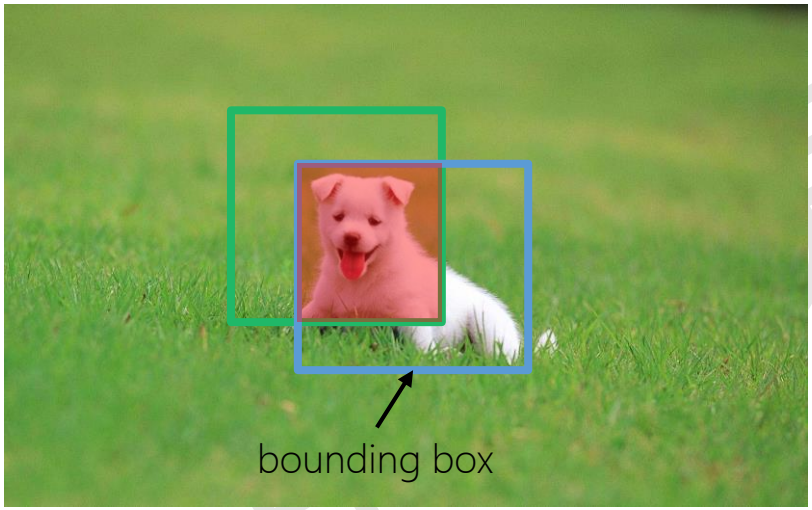
- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

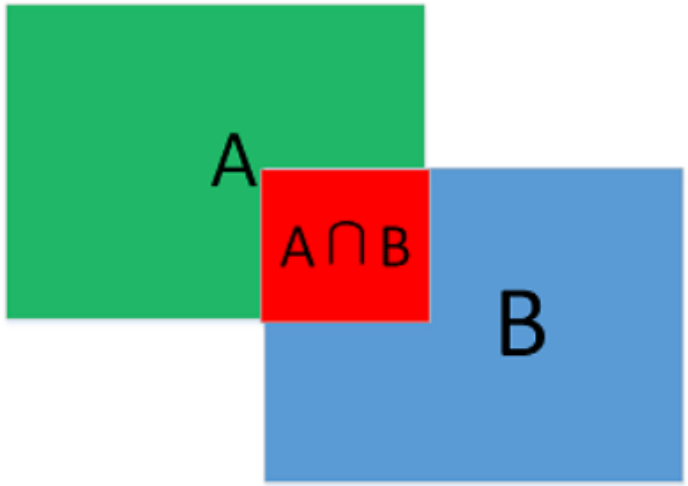
图像检测算法

	分类	定位+分类	物体检测
图示	 CAT	 CAT	 CAT, DOG, DUCK
输入	single and big object	single and big object	multi and small object
输出	label	label & bounding box	multi label & bounding box
评价	precision(top1/top5)	**IoU**(交并比)	**mAP**(Mean Average Precision)

评测指标——IoU

- IoU(交并比)
- 用于衡量定位准确度，一般 IoU ≥ 0.5 可认为定位成功 (true detection)；





$$IOU = \frac{A \cap B}{A \cup B}$$

评测指标——mAP

- mAP(mean Average Precision平均精度均值)
  - 在计算机视觉领域，用于衡量模型在测试集上检测精度的优劣程度；
  - 综合考虑检测结果的召回率/查全率和精度/查准率，mAP值越高表示检测结果越好；
- mAP计算原理

> - 召回率/查全率(recall)：选的N个样本里选对的k个正样本占总的M个正样本的比例 k/M；
> - 精度/查准率(precision)：选的N个样本里选对的k个正样本比例 k/N；
> - 选择的样本数N越多，召回率越高，查准率越低；

召回率 Recall=k/M=TP/(TP+FN)

精度 Precision=k/N=TP/(TP+FP)

测试结果	实际情况	
	正例(Positive)	反例(Negative)
正例(Positve)	真正例(True positive, TP)	假正例(False positve, FP)
反例(Negative)	假反例(False negative, FN)	真反例(True positive, TP)

评测指标——mAP

- 假设一个图像检测任务，共有5种类别，有100张图像作为测试集；
- 假设对于类A，100张测试图像中共有25个事先人为将类别标记为A的框；
- 假设算法对100张测试图像共检测出20个分类为A的候选框；

编号	置信度	标签
1	0.35	0
2	0.15	0
3	0.92	1
4	0.03	0
5	0.24	1
6	0.10	0
7	0.78	1
8	0.01	0
9	0.47	0
10	0.09	1
11	0.69	0
12	0.43	0
13	0.26	0
14	0.35	1
15	0.11	0
16	0.07	0
17	0.45	0
18	0.16	1
19	0.32	0
20	0.52	1

编号	置信度	标签
3	0.92	1
7	0.78	1
11	0.69	0
20	0.52	1
9	0.47	0
17	0.45	0
12	0.43	0
1	0.35	0
14	0.35	1
19	0.32	0
13	0.26	0
5	0.24	1
18	0.16	1
2	0.15	0
15	0.11	0
6	0.10	0
10	0.09	1
16	0.07	0
4	0.03	0
8	0.01	0

评测指标——mAP

例如：

confidence_threshold=0.5时：
第3, 7, 11, 20号框(score>0.5)被认为是positive，实际只有3, 7, 20号框(label=1)是true positive，那么此时precision=3/4，又因为总共应该有25个类别为A的框，那么recall=3/25

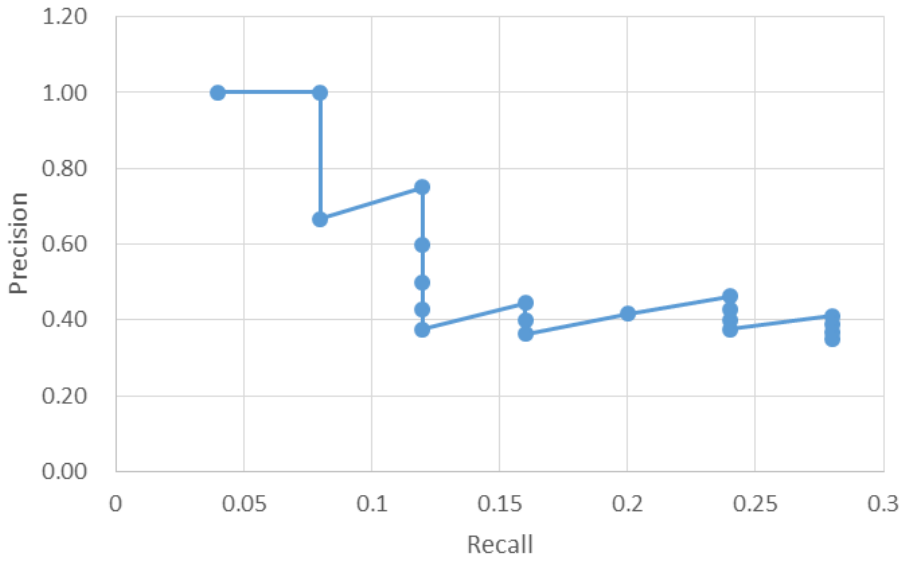
confidence_threshold=0.2时：
共有12个框(score>0.2)被认为是positive，实际只有5个框是true positive，此时 precision=5/12，recall=5/25

阈值越小，选中样本越多，精度越低，召回率越高

编号	置信度	标签
3	0.92	1
7	0.78	1
11	0.69	0
20	0.52	1
9	0.47	0
17	0.45	0
12	0.43	0
1	0.35	0
14	0.35	1
19	0.32	0
13	0.26	0
5	0.24	1
18	0.16	1
2	0.15	0
15	0.11	0
6	0.10	0
10	0.09	1
16	0.07	0
4	0.03	0
8	0.01	0

评测指标——mAP

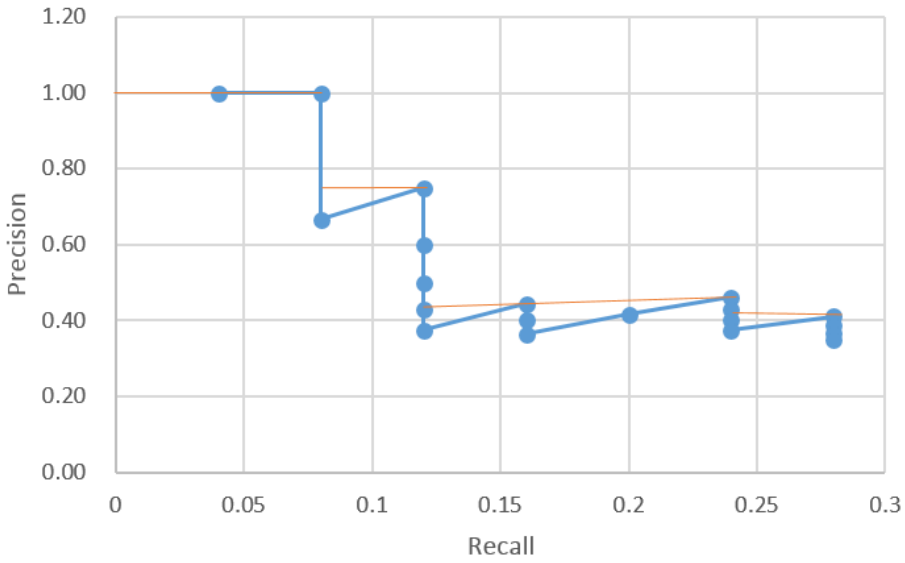
编号	置信度	标签	recall	precision
3	0.92	1	1/25	1/1
7	0.78	1	2/25	2/2
11	0.69	0	2/25	2/3
20	0.52	1	3/25	3/4
9	0.47	0	3/25	3/5
17	0.45	0	3/25	3/6
12	0.43	0	3/25	3/7
1	0.35	0	3/25	3/8
14	0.35	1	4/25	4/9
19	0.32	0	4/25	4/10
13	0.26	0	4/25	4/11
5	0.24	1	5/25	5/12
18	0.16	1	6/25	6/13
2	0.15	0	6/25	6/14
15	0.11	0	6/25	6/15
6	0.10	0	6/25	6/16
10	0.09	1	7/25	7/17
16	0.07	0	7/25	7/18
4	0.03	0	7/25	7/19
8	0.01	0	7/25	7/20



- 类A的AP值计算方法(VOC2012)：对每个recall值，取最大的precision求平均；
- 例如：recall=4/25时, 取precision=6/13；recall=6/25时, 取precision=6/13；

评测指标——mAP

编号	置信度	标签	recall	precision	maxPreci
3	0.92	1	1/25	1/1	1
7	0.78	1	2/25	2/2	1
11	0.69	0	2/25	2/3	
20	0.52	1	3/25	3/4	3/4
9	0.47	0	3/25	3/5	
17	0.45	0	3/25	3/6	
12	0.43	0	3/25	3/7	
1	0.35	0	3/25	3/8	
14	0.35	1	4/25	4/9	6/13
19	0.32	0	4/25	4/10	
13	0.26	0	4/25	4/11	
5	0.24	1	5/25	5/12	
18	0.16	1	6/25	6/13	
2	0.15	0	6/25	6/14	
15	0.11	0	6/25	6/15	
6	0.10	0	6/25	6/16	
10	0.09	1	7/25	7/17	7/17
16	0.07	0	7/25	7/18	
4	0.03	0	7/25	7/19	
8	0.01	0	7/25	7/20	



- 类A的AP值计算方法(VOC2012)：对每个recall值，取最大的precision求平均；
- 例如：recall=4/25时, 取precision=6/13；recall=6/25时, 取precision=6/13；
- AP(类A) = (1+1+3/4+6/13+6/13+6/13+7/17)/25=0.1819

基于CNN的图像检测算法

- R-CNN系列
- YOLO
- SSD

目前，基于深度学习的目标检测算法大致分为两类：

1.两阶段（two-stage）算法：基于候选区域方法，先产生边界框，再做CNN分类(R-CNN系列)

2.一阶段（one-stage）算法：**对输入图像**直接处理，同时输出定位及其类别 (YOLO系列)

R-CNN系列

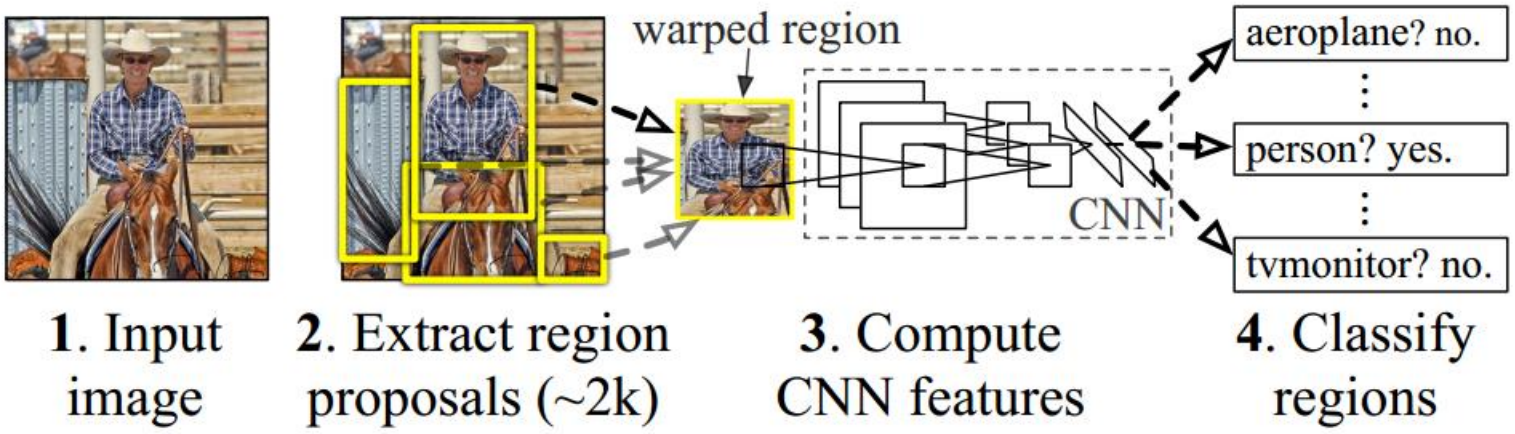
- R-CNN：Girshick R , Donahue J , Darrell T , et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]. 2014.
- Fast R-CNN：Girshick R. Fast R-CNN[J]. Computer Science, 2015.
- Faster R-CNN ： Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks[C], 2015:91-99.

网络	主要特点	mAP(VOC2012)	单帧检测时间
R-CNN	结合RegionProposal区域提取和CNN特征提取；SVM分类，Bounding Box回归；	53.3%	50 s
Fast R-CNN	提出 ROIPooling；softmax分类；	65.7%	2 s
Faster R-CNN	使用RPN(Region Proposal Network)生成候选区域	67.0%	0.2 s

R-CNN

R-CNN的主要步骤：

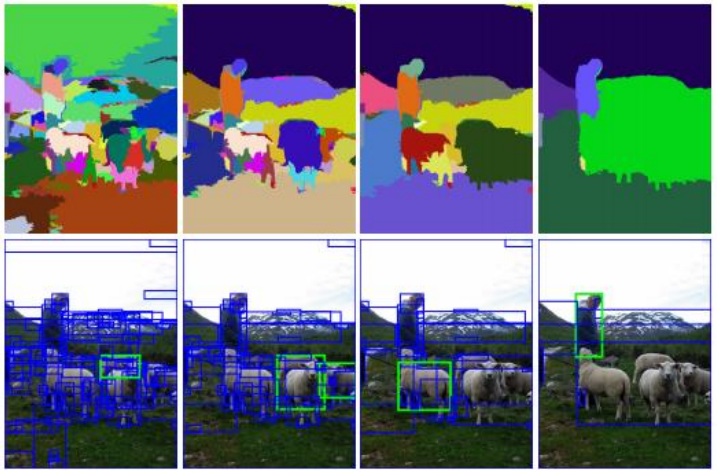
- **候选区域提取**：使用Selective Search从输入图片中提取2000个左右候选区域
- **特征提取**：首先将所有侯选区域裁切缩放为固定大小，再用AlexNet(5conv+2FC)提取图像特征
- **线性分类**：用特定类别的线性SVMs对每个候选区域做分类
- **边界框回归**：用线性回归修正边界框的位置与大小，其中每个类别单独训练一个边界框回归器



R-CNN

- 候选区域(Region Proposal)

- Uijlings J R R , K. E. A. van de Sande.... Selective Search for Object Recognition[J]. International Journal of Computer Vision, 2013, 104(2):154-171.
- **意义**：经典的目标检测算法使用滑动窗法依次判断所有可能的区域(穷举)，而 R-CNN 采用 Region Proposal 预先提取一系列较可能是物体的候选区域，之后仅在这些候选区域上提取特征，大大减少了计算量。
- **方法**：带多样性策略的**选择性搜索**(Selective Search)



R-CNN

- 候选区域提取 步骤

Algorithm 1: Hierarchical Grouping Algorithm

DontPrintSemicolon **Input**: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using Felzenszwalb and Huttenlocher (2004) Initialise similarity set $S = \emptyset$;
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$;
  $S = S \cup s(r_i, r_j)$;

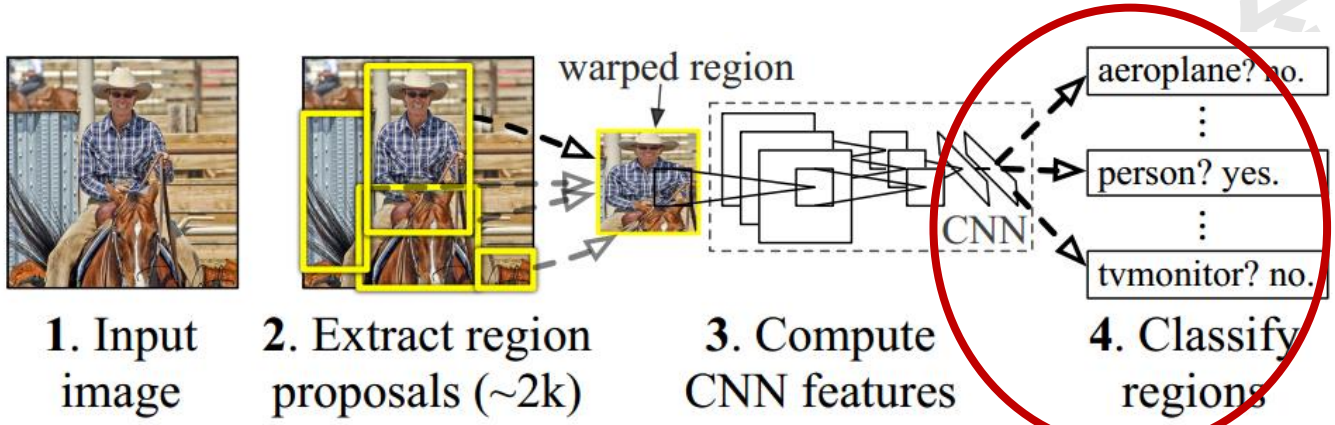
while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$;
  Merge corresponding regions $r_t = r_i \cup r_j$;
  Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$;
  Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$;
  Calculate similarity set S_t between r_t and its neighbours;
  $S = S \cup S_t$;
  $R = R \cup r_t$;
Extract object location boxes L from all regions in R;

- 层次化分组算法
  - 用基于图的图像分割方法创建初始区域
  - 计算所有相邻区域间的相似度
  - 每次合并相似度最高的两个相邻图像区域，并计算合并后的区域与其相邻区域的相似度。重复该过程，直到所有图像区域合并为一张完整图像
  - 提取所有图像区域的目标位置框，并按层级排序（覆盖整个图像的区域的层级为1）
- 在不同图像分割阈值、不同色彩空间、以及不同的相似度（综合考虑颜色、纹理、大小、重叠度）下，调用层次化分组算法，对所有合并策略下得到的位置框按层级*RND排序，去掉冗余框
- 取一定个数的候选区域作为后续卷积神经网络的输入(R-CNN取2000个)

Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T. & Smeulders, A. W. M. Selective Search for Object Recognition. International Journal of Computer Vision, 2013, 104, 154-171.

R-CNN

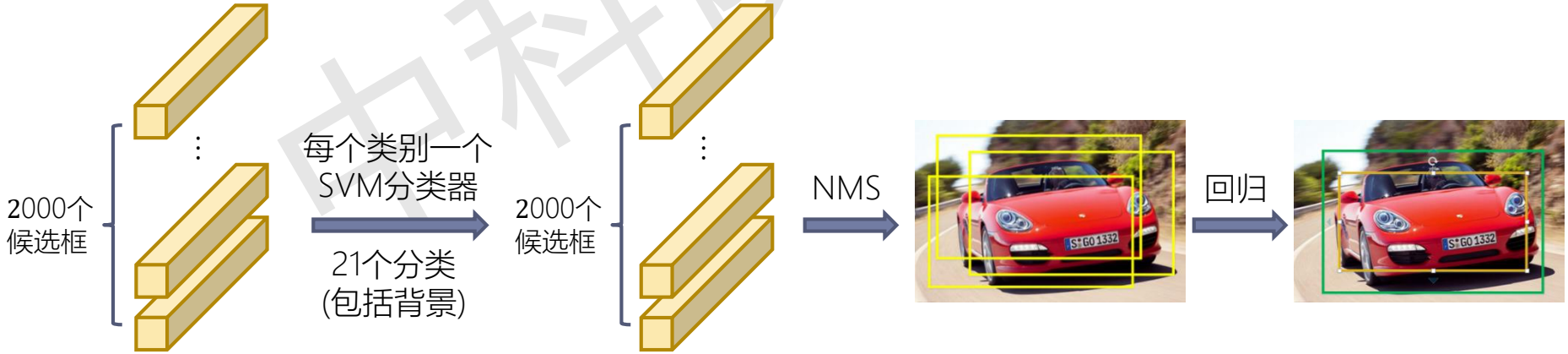
- 分类与回归



1. SVM分类

2. 非极大值抑制

3. BoundingBox回归



R-CNN

- 非极大值抑制(Non-Maximum Suppression, NMS)

- **问题**：同一目标的位置可能产生多个候选框，而这些候选框之间可能会有重叠
- **解决思路**：
  - 对每个分类，使用贪心NMS选择较优的目标边界框，去除冗余的边界框
- **NMS算法流程**：
  1. 按照检测得分对候选框排序
  2. 将分数最高的候选框b_m加到最终输出列表中，并将其从候选框列表中删除
  3. 计算b_m与其它候选框b_i的IoU；如果IoU大于阈值，则从候选框列表中删除b_i；
  4. 重复上述步骤，直至候选框列表为空。

R-CNN

R-CNN的缺点：

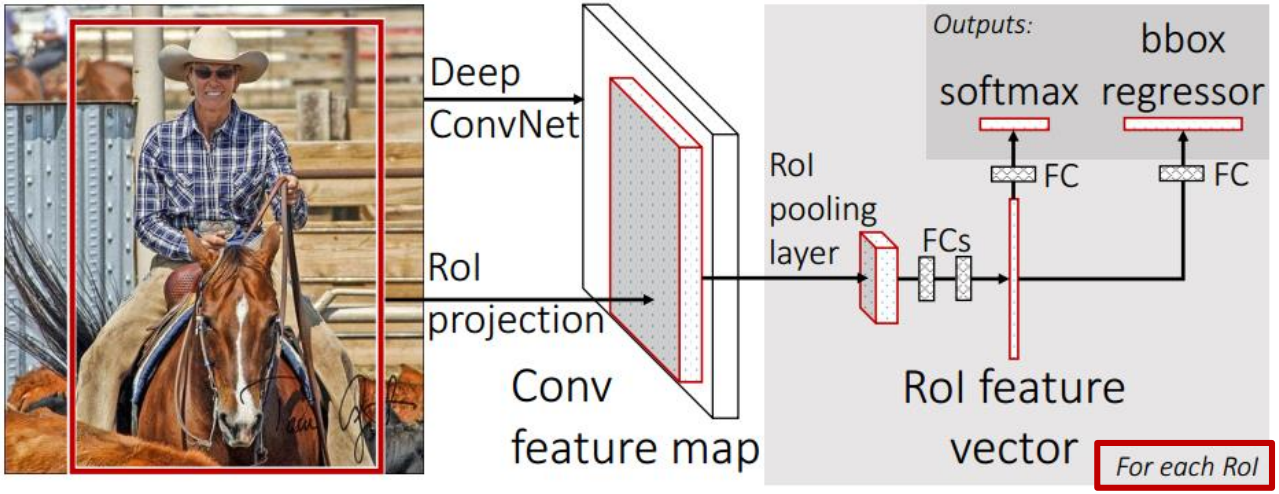
- 重复计算：需要对两千个候选框做CNN，计算量很大，而且有很多重复计算
- SVM模型：在标注数据足够的时候不是最好的选择
- 多个步骤：候选区域提取、特征提取、分类、回归都要单独训练，大量中间数据需要保存
- 检测速度慢：GPU上处理一张图片需要13秒，CPU上则需要53秒

能否避免候选框特征提取过程的重复计算？

Fast R-CNN

Fast R-CNN的主要步骤：

- **候选区域提取**：通过Selective Search从原始图片提取2000个左右区域候选框；
- **特征提取**：原始图像输入CNN网络，得到特征图；
- ROI-Pooling：根据映射关系，将不同尺寸的候选框在特征图上的对应区域池化为维度相同的特征图(因为全连接层要求输入尺寸固定)；
- **全连接层**：将维度相同的特征图转化为ROI特征向量(ROI feature vector)；
- **分类与回归**：经过全连接层，再用softmax分类器进行识别，用回归器修正边界框的位置与大小，最后对每个类别做NMS。



Fast R-CNN

- ROI Pooling

- **ROI**：regions of interest，对应前文中经过region proposal得到的的候选框；
- **目的**：将不同尺寸的ROI对应的卷积特征图转换为固定大小的特征图。一方面ROI可以复用卷积层提取的特征图，提高图像处理速度；另一方面向全连接层提供固定尺寸的特征图。
- **特点**：输出尺寸与输入尺寸无关。对每个特征图通道，根据输出尺寸(HxW)将输入(hxw)均分成多块(h/H x w/W大小/块），取每块的最大值作为输出。

Fast R-CNN

Fast R-CNN 改进之处：

- 直接对整张图像做卷积，不再对每个候选区域分别做卷积，从而减少大量的重复计算。
- 用ROI pooling对不同候选框的特征进行尺寸归一化。
- 将边界框回归器放进网络一起训练，每个类别对应一个回归器；
- 用softmax代替SVM分类器。

Fast R-CNN 缺点：

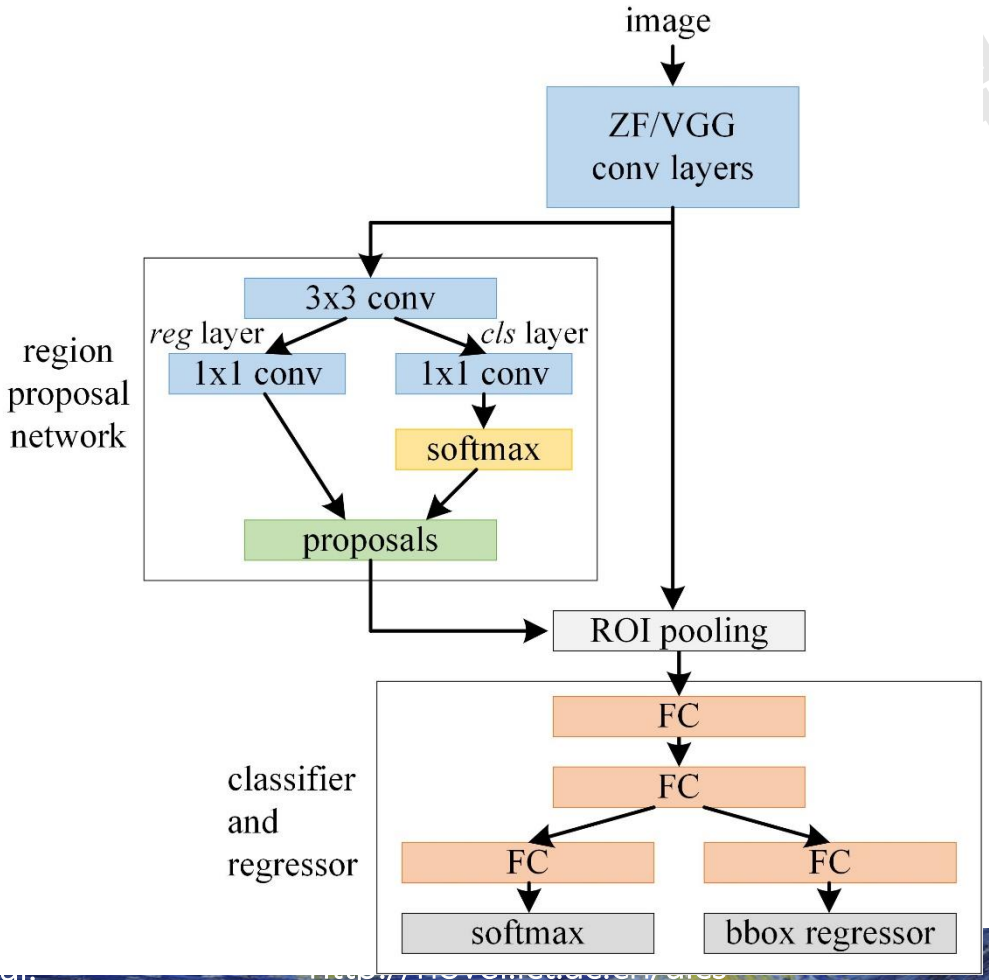
- 候选区域提取仍使用selective search，目标检测时间大多消耗在这上面（region proposal 2~3s，而特征分类只需0.32s）；

寻找更高效的候选区域生成方法?

Faster R-CNN

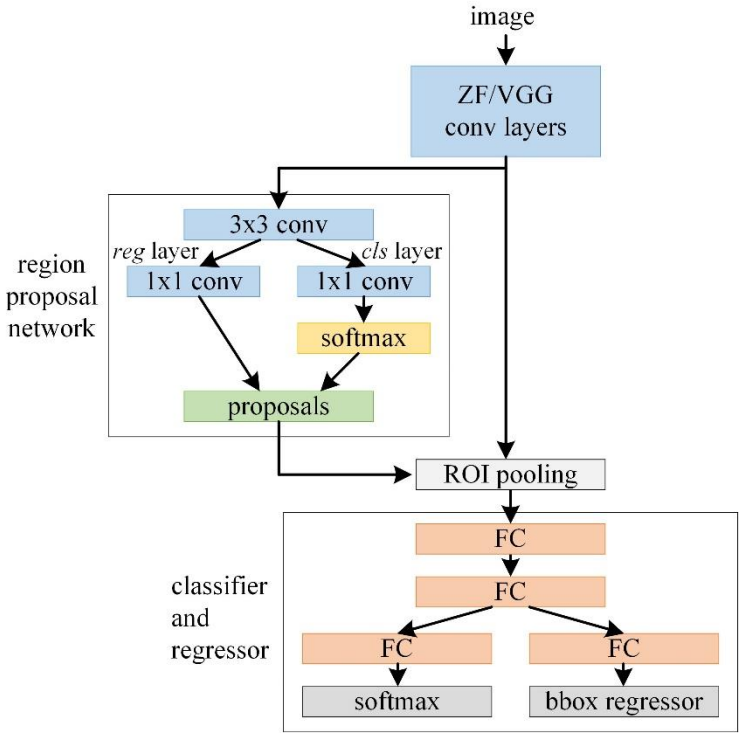
Faster R-CNN网络结构：

Faster R-CNN = 候选区域生成网络 RPN + Fast R-CNN



Faster R-CNN

Faster R-CNN主要步骤：



1.卷积层：输入图片经过多层卷积神经网络(ZF、VGG），提取出卷积特征图，供RPN网络和Fast R-CNN使用。RPN网络和Fast R-CNN共享特征提取网络可大大减小计算时间；

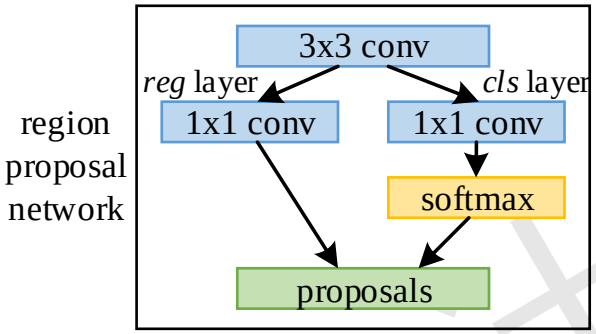
2. RPN层：生成候选区域，并用 softmax 判断候选框是前景还是背景，从中选取前景候选框并利用 bounding box regression 调整候选框的位置，得到候选区域；

3. ROIPooling层：同 Fast R-CNN，将不同尺寸的候选框在特征图上的对应区域池化为维度相同的特征图

4. 分类与回归：同 Fast R-CNN，用softmax分类器判断图像类别，同时用边界框回归修正边界框的位置和大小

Faster R-CNN

- RPN (region proposal networks)

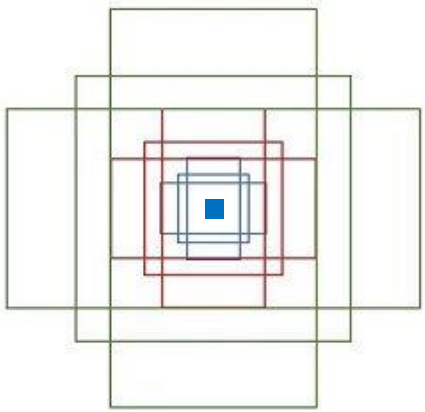


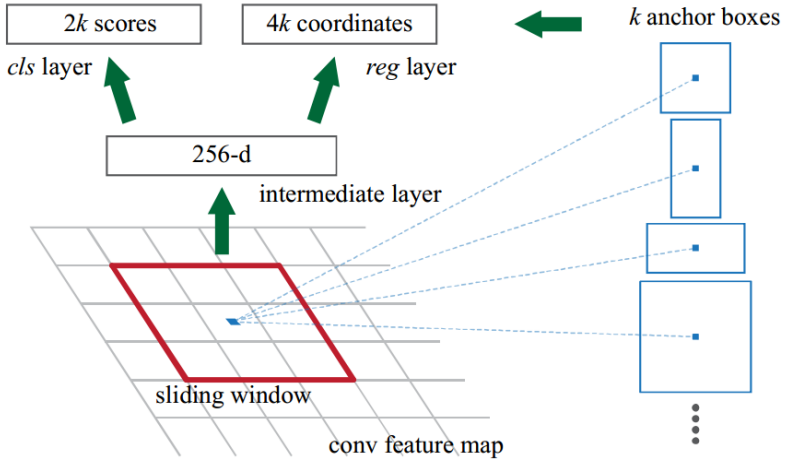
- **目的**：输入特征图，输出候选区域集合，包括各候选区域属于前/背景的概率、以及位置坐标。RPN采用Anchor机制能够从特征图上直接提取候选区域的特征，相对于selective search大大减少运算量，且整个过程融合到一个网络中，方便训练和测试。
- **步骤及方法**：

1. 先经过一个3x3卷积，使每一个点对应256维(ZF模型）或512维（VGG16）特征向量。
2. 然后分两路处理：一路用来判断候选框是前景还是背景，先将卷积特征reshape成一维向量，然后做softmax判断是前景还是背景，然后reshape回二维feature map；另一路用bbox regression来确定候选框的位置。
3. 两路计算结束后，计算得到前景候选框（因为物体在前景中），再用NMS去除冗余候选框，最后输出候选区域。

Faster R-CNN

- 关于anchor box



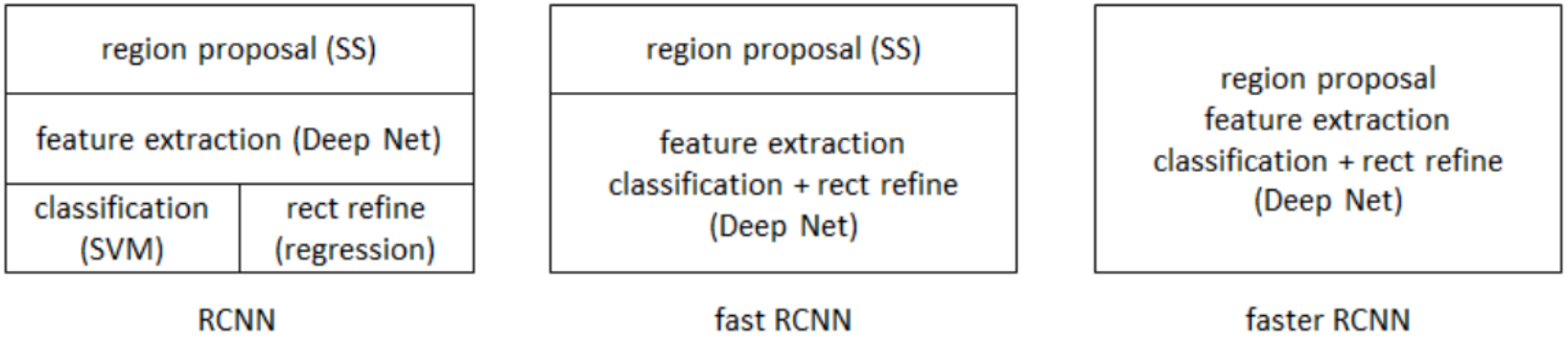


- 对于feature map的每个位置，考虑9个可能的候选框：三种面积分别是128×128, 256×256, 512×512，每种面积又分成3种长宽比，分别是2:1, 1:2, 1:1，这些候选框称为anchors。

- 在RPN中，feature map每个位置输出2k个得分，分别表示该位置的k个anchor为前景/背景的概率；同时每个位置输出4k个框位置参数，用 [x,y,w,h] 四个坐标来表示 anchor的位置。

R-CNN 系列

- R-CNN系列



网络	mAP (VOC2012)	单帧检测时间
R-CNN	53.3%	50 s
Fast R-CNN	65.7%	2 s
Faster R-CNN	67.0%	0.2 s

- 从R-CNN到Fast R-CNN，再到Faster R-CNN，目标检测的四个基本步骤（候选区域生成，特征提取，分类，位置调整）终于被统一到一个深度网络框架之内，大大提高了运行速度。

拓展：R-CNN训练相关，Ross Girshick在ICCV15的演讲，Training R-CNNs of various velocities(Slow, fast, and faster)

基于CNN的图像检测算法

- R-CNN系列
- YOLO
- SSD

One-stage

目前，基于深度学习的目标检测算法大致分为两类：

1.两阶段（two-stage）算法：基于候选区域方法，先产生边界框，再做CNN分类(R-CNN系列)

2.一阶段（one-stage）算法：**对输入图像**直接处理，同时输出定位及其类别 (YOLO系列)

YOLO

- YOLO(v1)：Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2015.
- **主要思想**：将目标检测问题转换为直接从图像中提取bounding boxes和类别概率的**单回归问题**，只需看一眼(you only look once, YOLO) 就可以检测出目标的类别和位置。YOLO算法开创了one-stage检测的先河，将目标分类和边界框定位合二为一，实现了端到端的目标检测。YOLO的运行速度非常快，达到45帧/秒，满足实时性要求。

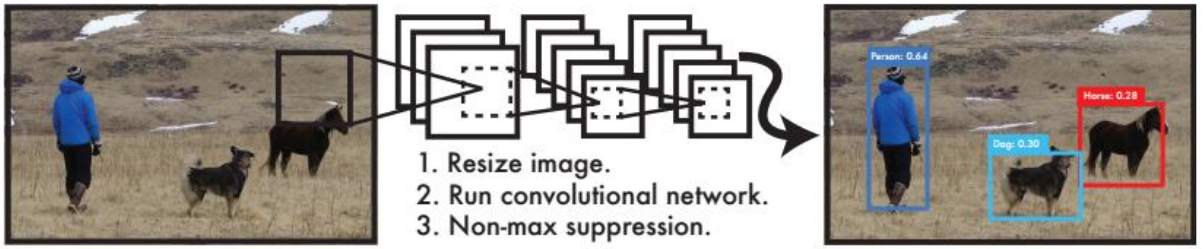
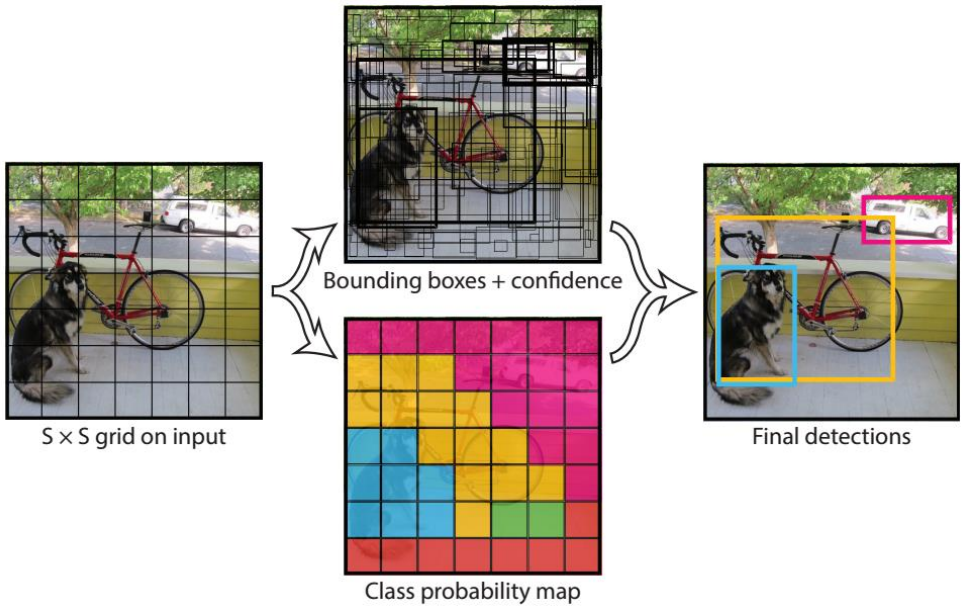


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

YOLO

- 统一检测(Unified Detection)具体实现



- 将输入图像分为S×S个格子，每个格子都预测B个Bounding box，每个bbox包含五个预测值：x, y, w, h和confidence;
- x, y, w, h用于表示bbox的位置和大小，且都被归一化到(0, 1);
- confidence(置信度分数) 综合考虑了当前bbox内存在目标的可能性Pr(Object)以及预测目标位置的准确性IOU(pred|truth)，定义为：

$$confidence = \Pr(Object) * IOU_{pred}^{truth}, \quad \Pr(Object) = \begin{cases} 1, & object\ exists\ in\ that\ cell \\ 0, & no\ object\ exists\ in\ that\ cell \end{cases}$$

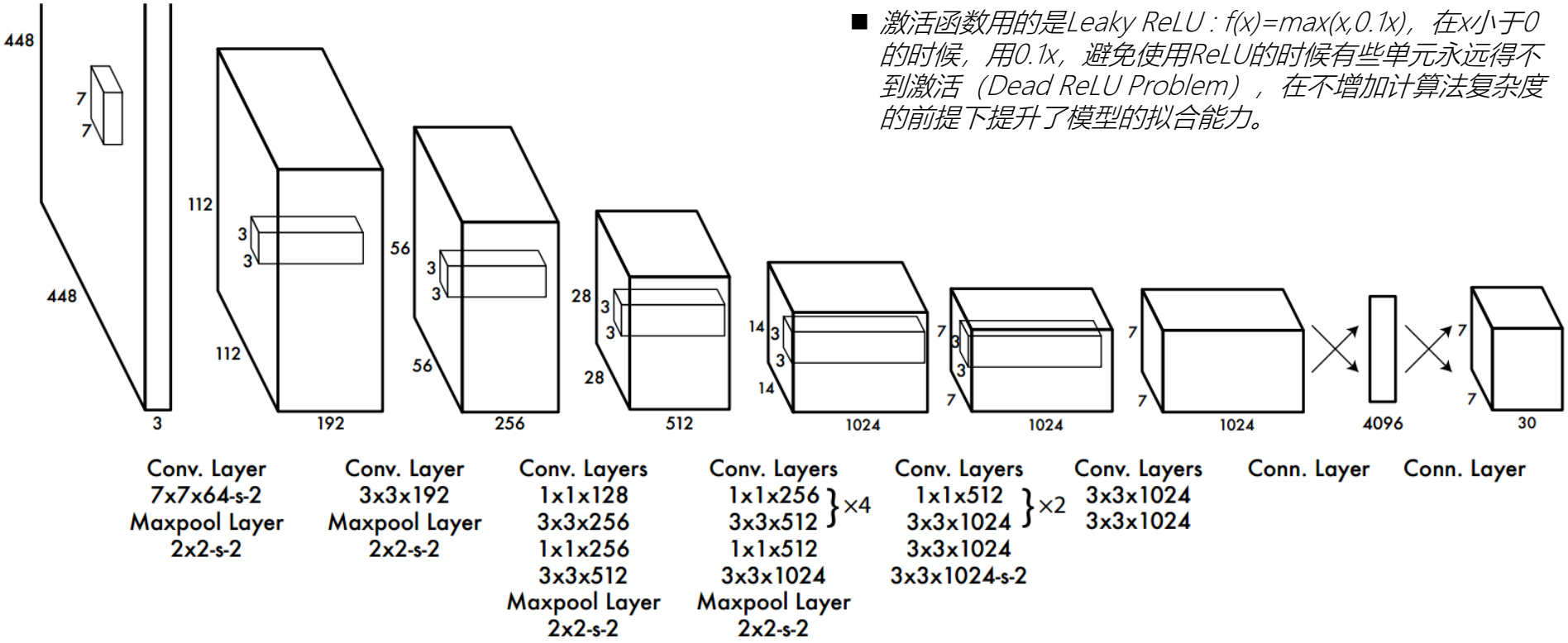
- 每个格子还要预测分别属于C种类别的条件概率$\Pr(Class_i|Object), i = 0,1,\dots C$，其中C为数据集物体类别数量；在测试时，属于该格子的B个bbox共享这C个类别的条件概率，则某个bbox的类别置信度为：

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- 最终输出tensor的维度为：$S \times S \times (B \times 5 + C)$

YOLO

- 网络结构



- *激活函数用的是Leaky ReLU : f(x)=max(x,0.1x)，在x小于0的时候，用0.1x，避免使用ReLU的时候有些单元永远得不到激活（Dead ReLU Problem），在不增加计算法复杂度的前提下提升了模型的拟合能力。*

- 网络结构基于GoogleNet；
- 对于PASCAL VOC数据集，采用S=7， B=2， C=20， 最终输出tensor维度为 7×7×30 (其中30=B*5+C)。

YOLO

YOLO(v1)的优点

1、检测速度快。 YOLO将目标检测重建为单一回归问题，对输入图像直接处理，同时输出边界框坐标和分类概率，而且每张图像只预测98个bbox，检测速度非常快，在Titan X 的 GPU 上能达到45 FPS，Fast YOLO检测速度可以达到155 FPS。

2、背景误判少。以往基于滑窗或候选区域提取的目标检测算法，只能看到图像的局部信息，会出现把背景当前景的问题。而YOLO在回归之前做了全连接，在训练和测试时每个cell都使用全局信息做预测，因此不容易把背景误认为目标。

3、泛化性更好。 YOLO能够学习到目标的泛化表示，能够迁移到其它领域。例如，当YOLO在自然图像上做训练，在艺术品上做测试时，YOLO的性能远优于DPM、R-CNN等。

YOLO

YOLO(v1) 的缺点

1、邻近物体检测精度低。YOLO对每个cell只预测两个bbox和一个分类，如果多个物体的中心都在同一cell内，检测精度低。

2、损失函数的设计过于简单。用坐标和分类的MSE作为损失函数不合理

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

$\mathbb{1}_{i}^{\text{obj}}$ 表示目标出现在cell i中　　$\mathbb{1}_{ij}^{\text{obj}}$ 表示cell i中第j个边框预测目标在该cell中

3、训练不易收敛。直接预测的bbox位置，相较于预测物体的偏移量，模型收敛不稳定。

YOLO

- 拓展

- YOLO-v2 ： Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger[C]// IEEE Conference on Computer Vision & Pattern Recognition. 2017.

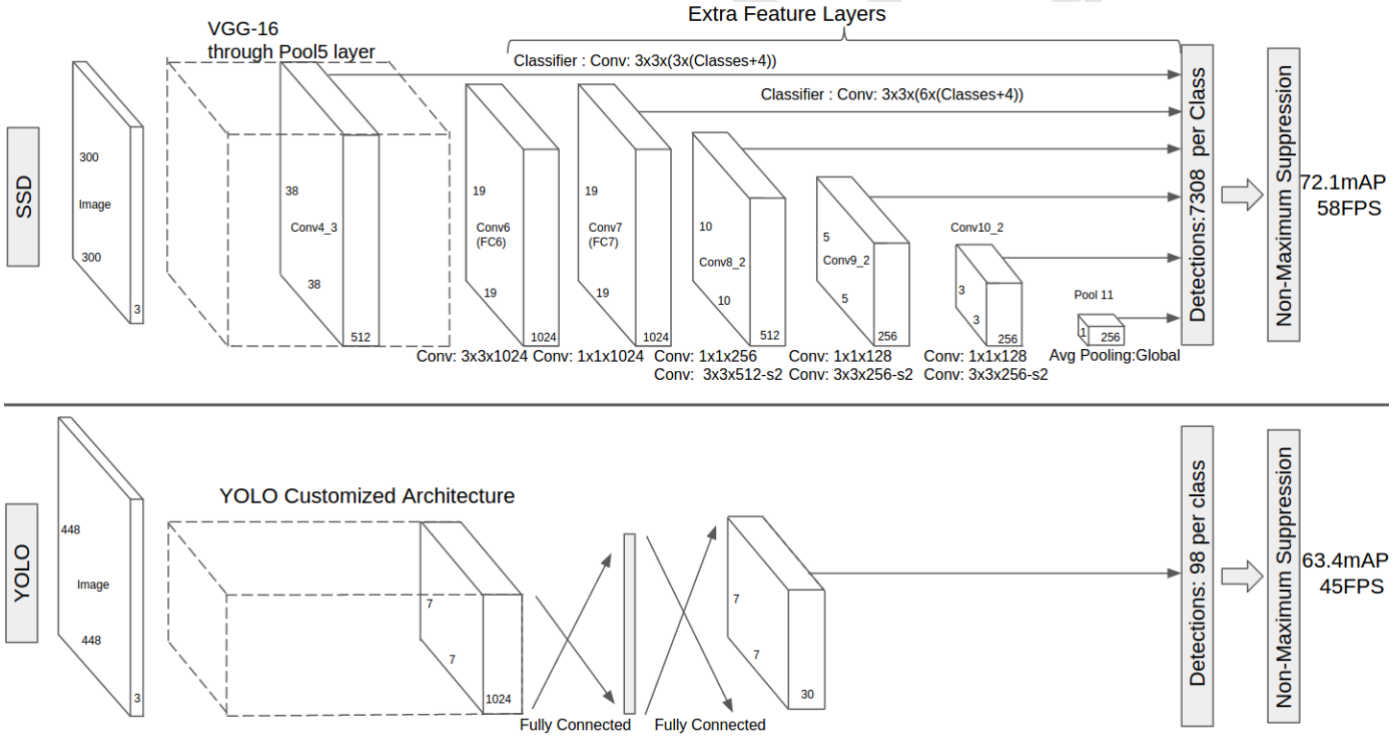
  提高了训练图像的分辨率；引入了faster rcnn中anchor box的思想；对网络结构的设计进行了改进(Darknet-19)；输出层使用卷积层替代YOLO的全连接层，联合使用coco物体检测标注数据和imagenet物体分类标注数据训练物体检测模型。

- YOLO-v3 ： Redmon J, Farhadi A. YOLOv3: An Incremental Improvement[J]. 2018.

  类似FPN的多尺度预测；更好的基础分类网络（Darknet-53，结合resnet）；Sigmoid代替Softmax用于多标记分类。

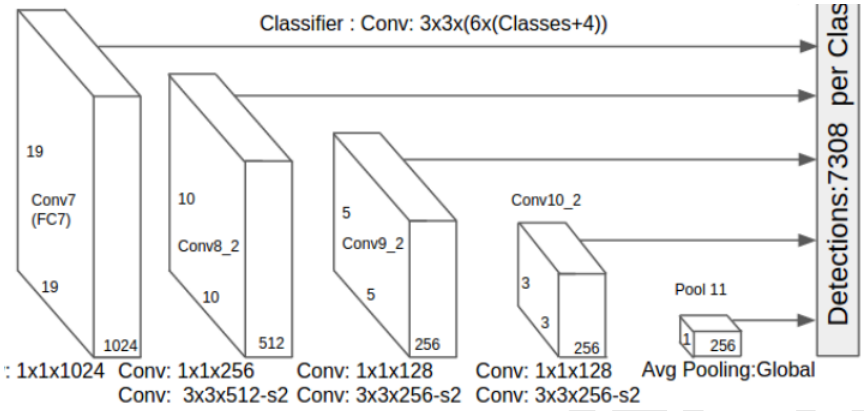
SSD

- Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[C]// European Conference on Computer Vision. 2016.
- **主要思想**：基于YOLO直接回归 bbox和分类概率的one-stage检测方法，结合Faster R-CNN中的anchor-box思想产生先验框，并且采用特征金字塔进行多尺度预测，在满足检测速度快的同时，大大提高了检测准确度。

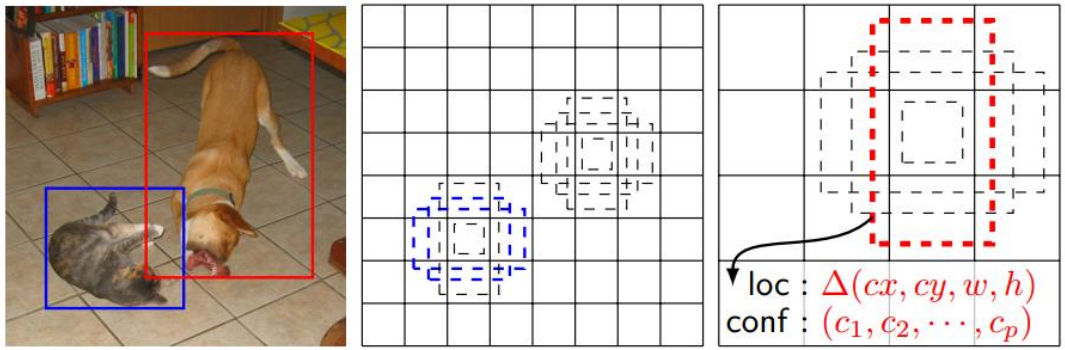


SSD

- 多尺度特征图检测



- CNN网络一般前面的特征图比较大，后面会逐渐采用stride=2的卷积或者pool来降低特征图大小，在大的和小的特征图都提取anchor box用来做检测，可找到最合适的anchor box尺寸，提高检测准确度。



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- 比较大的特征图可以用来检测相对较小的目标，而小的特征图负责检测大目标，例如左图中8x8的特征图可以划分成更多单元，其每个单元的先验框尺度较小，适合用于检测较小的目标。

SSD

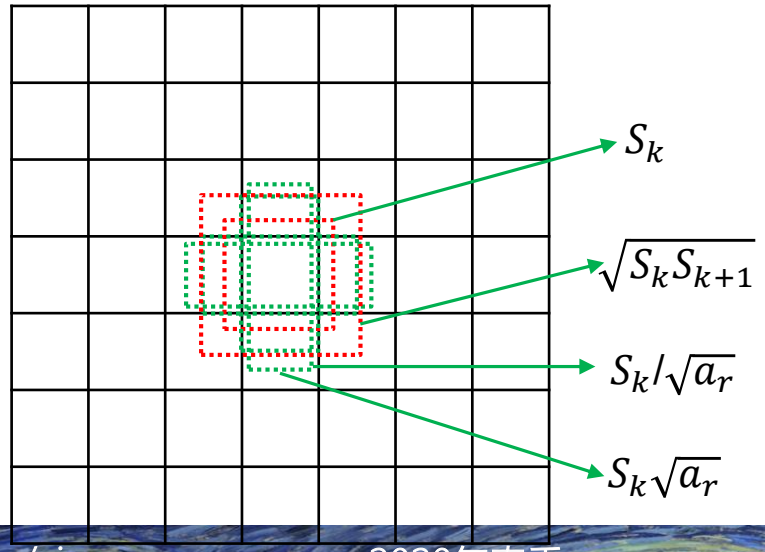
- Anchor box (论文中为default box)

- 个数：每个点对应6个anchor boxes，2个正方形和4个长方形(宽高比a_r取2、3、1/2、1/3)；
- 第k层scale： $s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m]$

  其中：$S_{min} = 0.2, S_{\max} = 0.9$，m为网络用于生成anchor box进行检测的卷积层总层数；

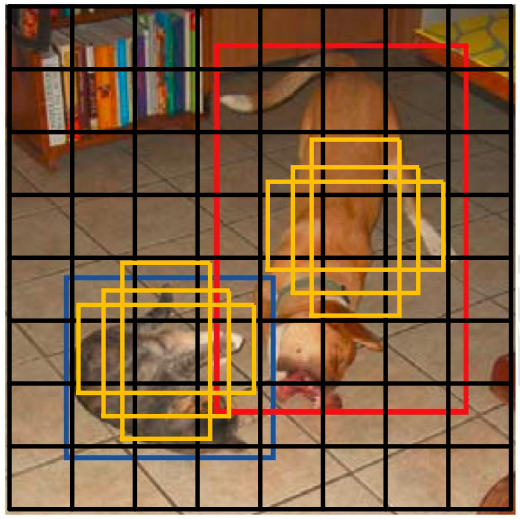
  宽高比为1的情况，增加一个默认框，其缩放为 $s'_k = \sqrt{s_k s_{k+1}}$
- 第k层default box的宽：$w_k^a = s_k\sqrt{a_r}$

  高：$h_k^a = s_k/\sqrt{a_r}$

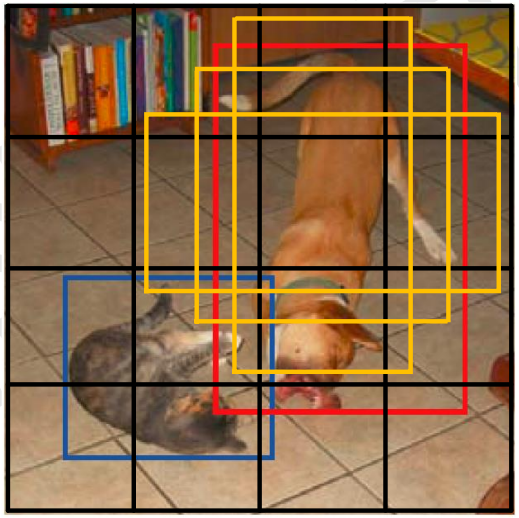


SSD

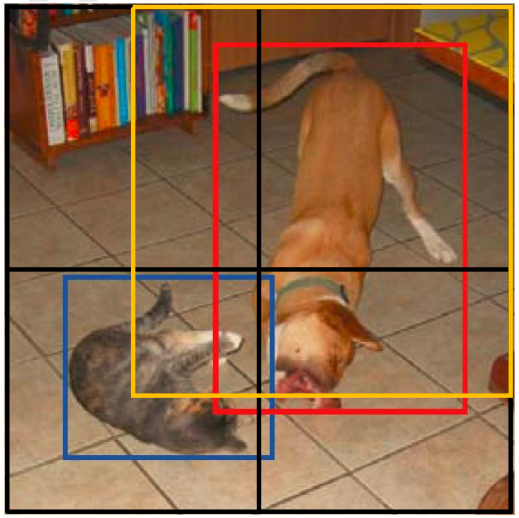
- 同时对多层特征图上的默认框计算IOU，可以找到与真实框大小和位置最接近（即IOU最大）的框，在训练时能达到最好的精度。



较低层级的特征图

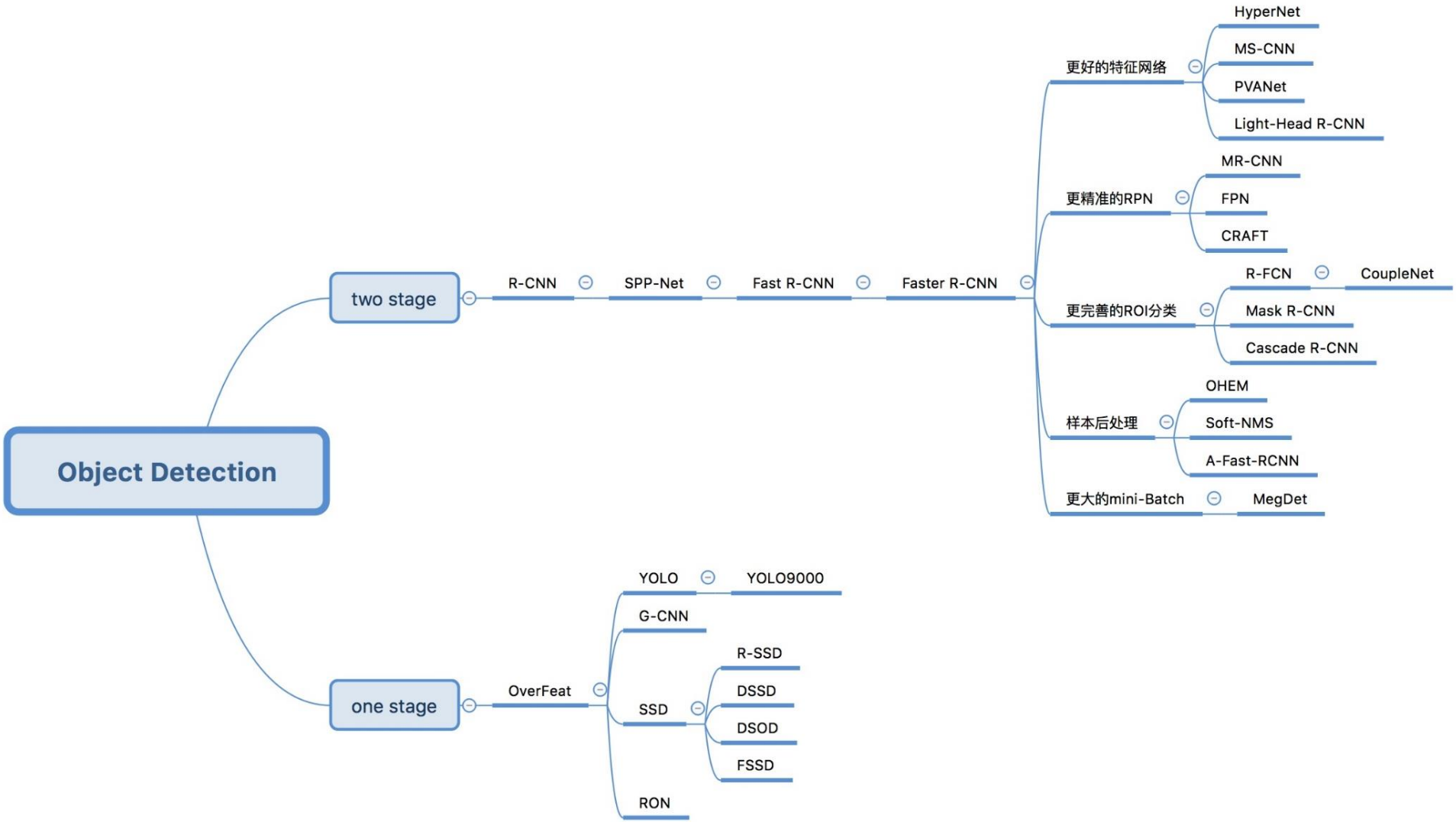


中间层级的特征图



较高层级的特征图

图像检测算法



https://zhuanlan.zhihu.com/p/33277354

提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

序列模型：循环神经网络

图像信息

任务：理解图像内容

方法：卷积神经网络

序列信息

任务：理解语音/文字/视频

方法：循环神经网络

任务特点：按时序输入的数据之间不是相互独立的，前后数据之间有相关性，所以网络要有"存储"信息的能力。

序列模型：循环神经网络

- 主要应用场景：机器翻译、图片描述、视频标注、视觉问答等

美国总统布什昨天在白宫与以色列总理沙龙就中东局势举行了一个小时的会谈。

US President George W. Bush held an hour-long meeting with Israeli Prime Minister Ariel Sharon on the situation in the Middle East yesterday at the White House.

机器翻译

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

图片描述

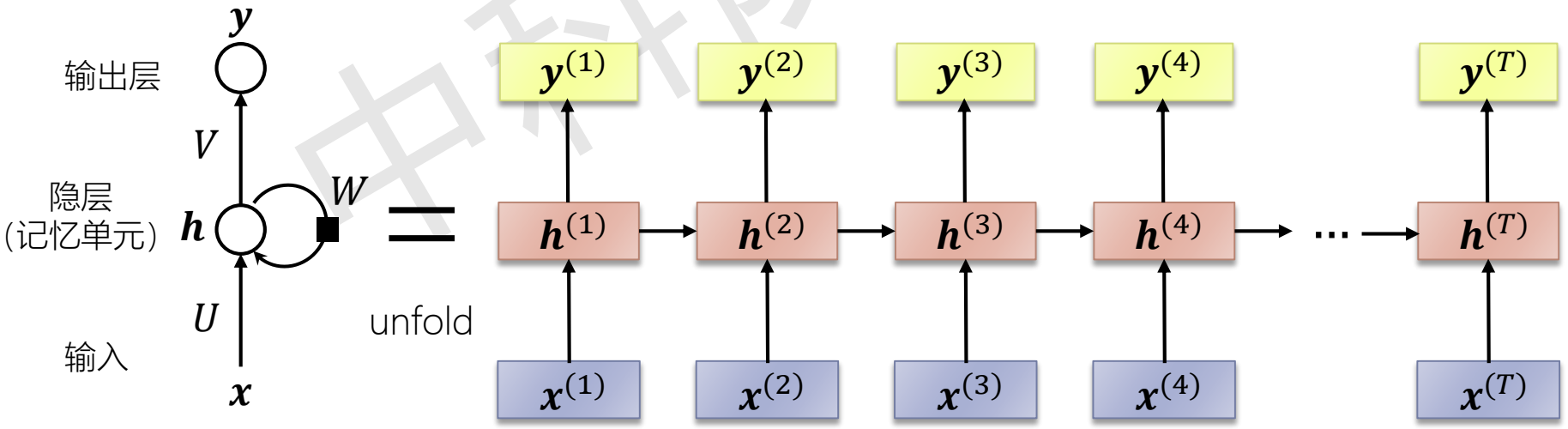
主要应用场景

- Applications
  - Natural Language Processing
    - Language Modeling
    - Speech Recognition
    - Machine Translation
    - Conversation Modeling
    - Question Answering
  - Computer Vision
    - Object Recognition
    - Image Generation
    - Video Analysis
  - Multimodal (CV+NLP)
    - Image Captioning
    - Video Captioning
    - Visual Question Answering
  - Turing Machines
  - Robotics
  - Other

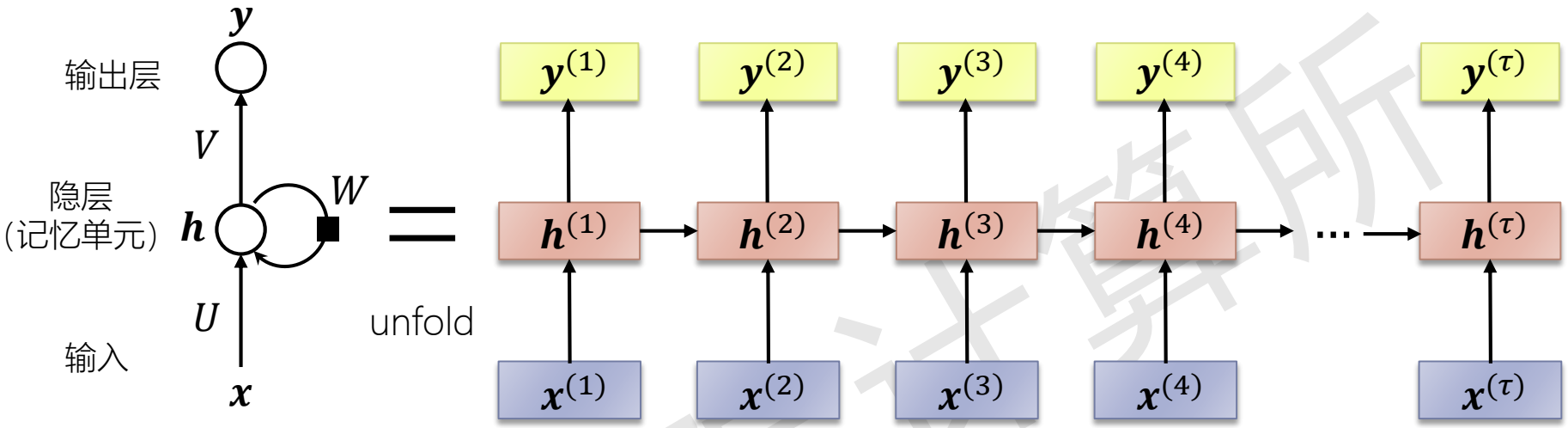
- GitHub Project: **Awesome Recurrent Neural Networks**
- A curated list of resources dedicated to recurrent neural networks (closely related to deep learning).
- https://github.com/kjw0612/awesome-rnn

循环神经网络结构

- 循环神经网络（RNN）通过使用带自反馈的神经元，能够处理任意长度的序列；
- 时刻t的输入为$\boldsymbol{x}^{(t)}$，隐藏状态为$\boldsymbol{h}^{(t)}$。$\boldsymbol{h}^{(t)}$既和当前时刻输入$\boldsymbol{x}^{(t)}$相关，也和上一时刻隐藏状态$\boldsymbol{h}^{(t-1)}$相关；
- $\boldsymbol{h}^{(t)} = f\left(W\boldsymbol{h}^{(t-1)} + U\boldsymbol{x}^{(t)} + \boldsymbol{b}\right)$，$f$是非线性激活函数，常用tanh或ReLU



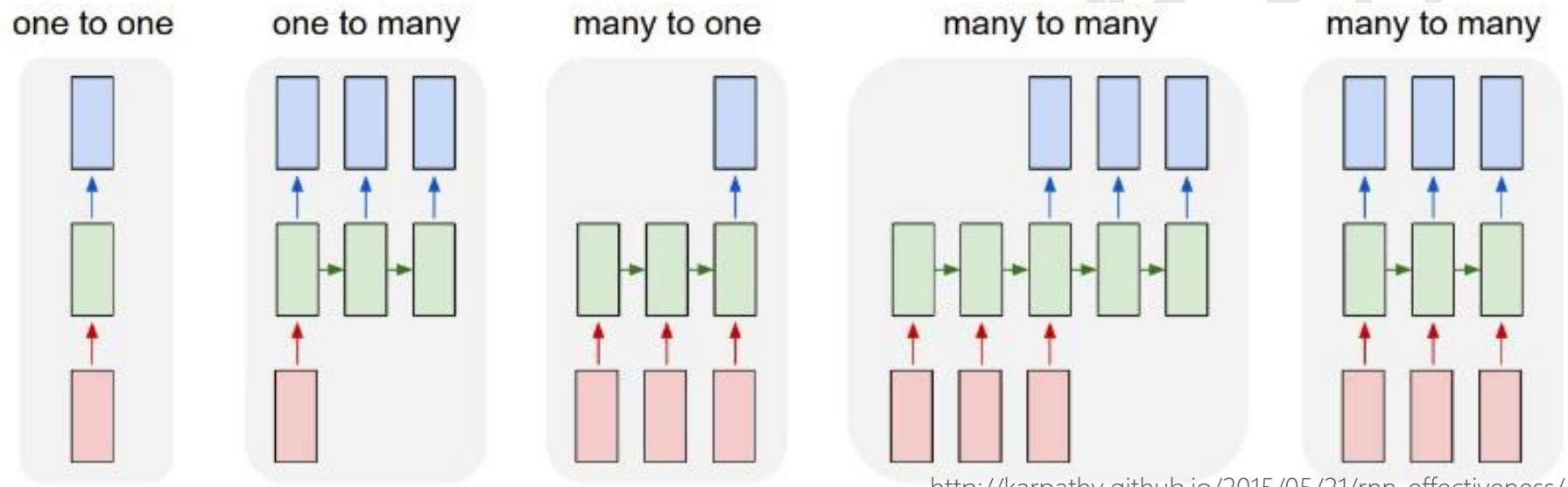
循环神经网络结构



- **时序**sequence：RNN能建模序列数据，序列指的是前、后输入数据($\boldsymbol{x}^{(t)}$, $\boldsymbol{x}^{(t+1)}$)不独立，相互影响；
- **循环**recurrent：对每个输入的操作都是一样的，循环往复地重复这些相同操作，每时刻有相同参数W和U（参数共享）；
- **记忆**memory： 隐藏层$\boldsymbol{h}^{(t)}$中捕捉了所有时刻t之前的信息，理论上$\boldsymbol{h}^{(t)}$记忆的内容可以无限长，然而实际上记忆还是有限的；

循环神经网络结构

- RNN的多种输入-输出结构



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

序列作为输出
image captioning

序列作为输入
sentiment analysis

序列转化为序列
machine translation
video caption

同步序列转化为序列
video classification
(标注每一帧)

对序列的长度（绿色块块多少个）无要求，不用预先定义

循环神经网络结构

- 正向计算过程

初始时刻：h_0 为输入的初始零向量

$$\boldsymbol{h}^{(1)} = f(W\boldsymbol{h}^{(0)} + U\boldsymbol{x}^{(1)} + \boldsymbol{b})$$

$$\boldsymbol{o}^{(1)} = V\boldsymbol{h}^{(1)} + \boldsymbol{c}$$

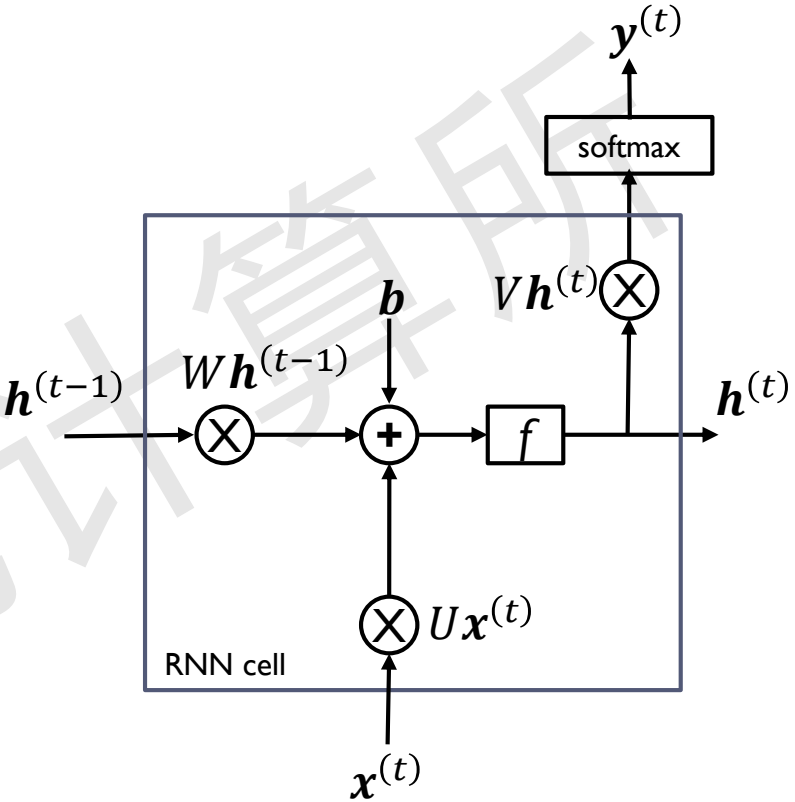
$$\hat{\boldsymbol{y}}^{(1)} = softmax(\boldsymbol{o}^{(1)})$$

一般时刻：

$$\boldsymbol{h}^{(t)} = f(W\boldsymbol{h}^{(t-1)} + U\boldsymbol{x}^{(t)} + \boldsymbol{b})$$

$$\boldsymbol{o}^{(t)} = V\boldsymbol{h}^{(t)} + \boldsymbol{c}$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax(\boldsymbol{o}^{(t)})$$



- f常用$\tanh$或$Relu$；
- W、U、V、$\boldsymbol{b}$ 在时间序列上共享参数；

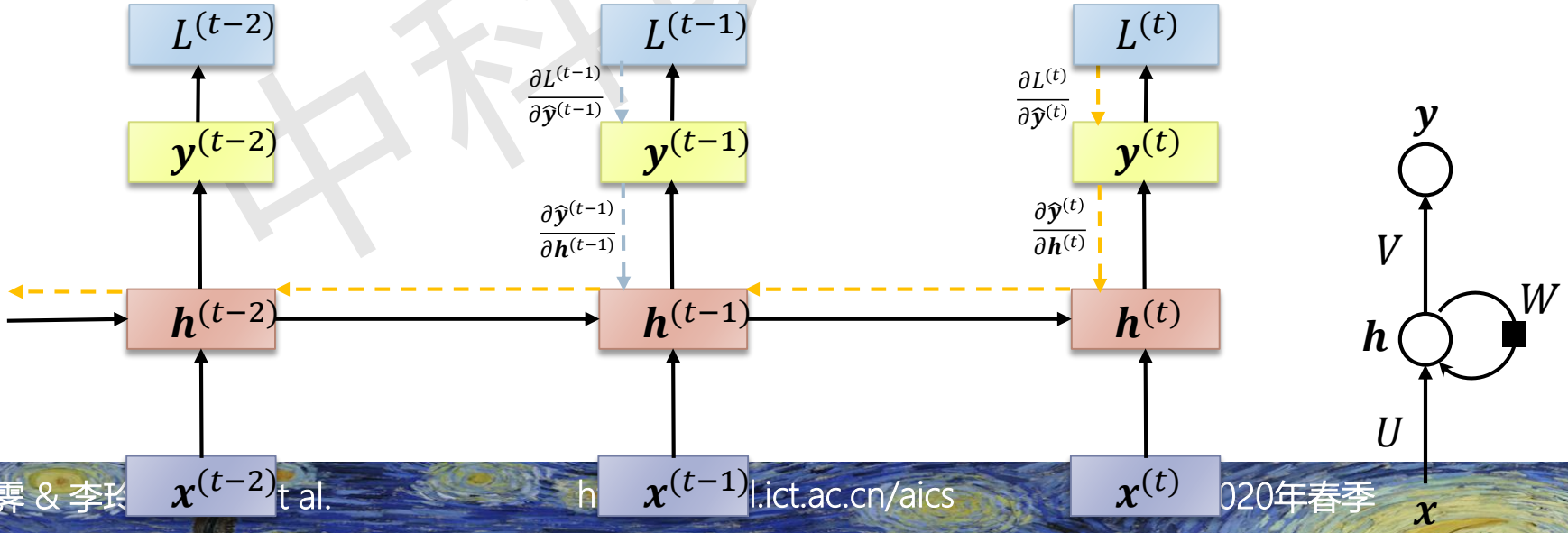
循环神经网络结构

- 反向传播BPTT (back-propagation through time)

某个时刻的损失函数为：$L^{(t)} = -\boldsymbol{y}^{(t)} \ln \hat{\boldsymbol{y}}^{(t)}$

整个序列的损失函数为：$L = \sum_{t=1}^{\tau} L^{(t)} = -\sum_{t=1}^{\tau} \boldsymbol{y}^{(t)} \ln \hat{\boldsymbol{y}}^{(t)}$

损失函数对W的偏导为：$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial J^{(t)}}{\partial \hat{\boldsymbol{y}}^{(t)}} \frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(k)}} \frac{\partial \boldsymbol{h}^{(k)}}{\partial W}$



RNN的梯度消失与梯度爆炸

- 梯度爆炸/梯度消失

根据推导可知序列损失函数对U和W的梯度为：

$$\frac{\partial L}{\partial W}=\sum_{t}\sum_{k=1}^{k=t}\frac{\partial L^{(t)}}{\partial \hat{\boldsymbol{y}}^{(t)}}\frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{o}^{(t)}}\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\left(\prod_{i=k+1}^{t}W^{\top}diag\left(1-(\boldsymbol{h}^{(i)})^{2}\right)\right)\frac{\partial \boldsymbol{h}^{(k)}}{\partial W}$$

$$\frac{\partial L}{\partial U}=\sum_{t}\sum_{k=1}^{k=t}\frac{\partial L^{(t)}}{\partial \hat{\boldsymbol{y}}^{(t)}}\frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{o}^{(t)}}\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\left(\prod_{i=k+1}^{t}W^{\top}diag\left(1-(\boldsymbol{h}^{(i)})^{2}\right)\right)\frac{\partial \boldsymbol{h}^{(k)}}{\partial U}$$

令 $\gamma=\|\prod_{i=k+1}^{t}W^{\top}diag\left(1-(\boldsymbol{h}^{(i)})^{2}\right)\|_{2}$

当$t\gg k$时，
$$\gamma\begin{cases}\to\infty, & \frac{\partial L}{\partial U}\to\infty, \quad \frac{\partial L}{\partial W}\to\infty \text{ 梯度爆炸}\\ \to 0, & \frac{\partial L}{\partial U}\to 0, \quad \frac{\partial L}{\partial W}\to 0 \text{ 梯度消失}\end{cases}$$

由于梯度爆炸或消失的存在，循环神经网络实际上只能学习到短期的依赖关系。

RNN的梯度消失与梯度爆炸

- 循环神经网络的递归结构导致梯度消失/梯度爆炸现象更明显

- 如何改进
  - 梯度爆炸问题
    - 梯度截断

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

Pascanu, R.; Mikolov, T. & Bengio, Y. Dasgupta, S. & McAllester, D. *(Eds.)* On the difficulty of training recurrent neural networks， ICML **2013**

  - 梯度消失问题
    - 模型上的改进，例如LSTM、GRU算法

RNN的梯度消失与梯度爆炸

- **由于梯度消失，循环神经网络无法处理长期依赖关系；**

- 比如，考虑一个语言模型，试图根据之前单词预测下一个。
- 如果要预测“The clouds are in the sky”中最后一个单词，不需要更多的上下文即可知道下一个单词会是“sky”。在这种情况下，相关信息与预测位置的间隔比较小，RNNs可以学会使用之前的信息。
- 考虑试图预测“I grew up in Italy… I speak fluent Italian.”中最后一个，则需要用到包含“Italy”的上下文，从前面的信息推断后面的单词。相关信息与预测位置的间隔可能会很大。随着这种间隔的拉长，RNNs就会无法学习连接信息。

如何记住长期依赖？

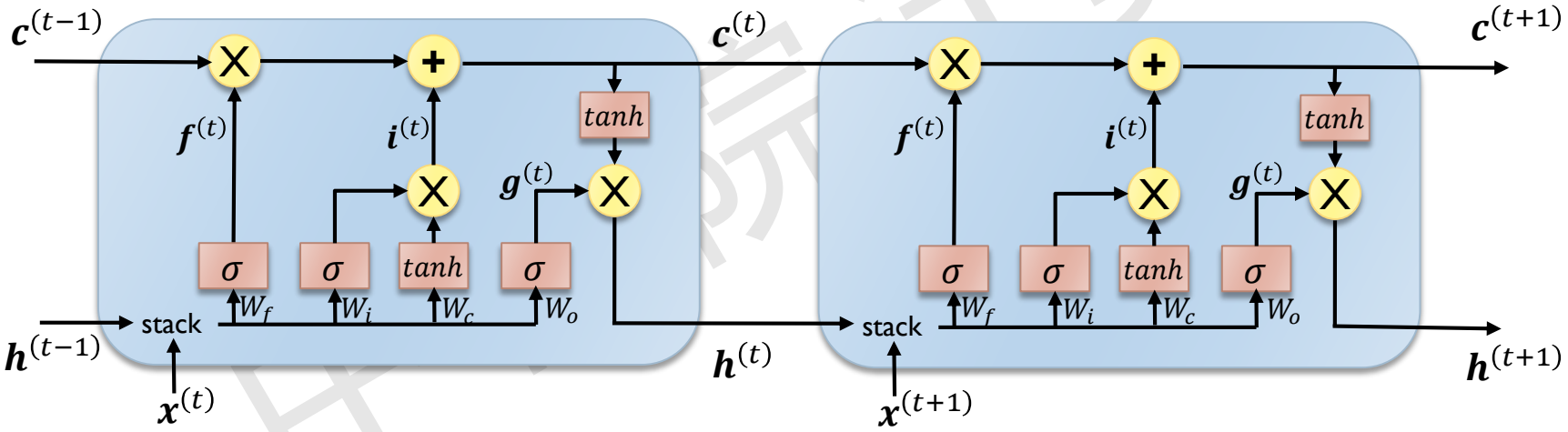
提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

长短期记忆模型

- LSTM (Long Short-Term Memory networks)

  - Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.



- 隐藏状态：作为神经网络的记忆，保存着网络先前观察到的数据信息。
- 单元状态：类似信息传送带，它贯穿整个链条，只有一些小的线性相互作用；这很容易让信息以不变的方式向下流动；LSTM有能力向单元状态中移除或添加信息，这种管理结构称为门限。

长短期记忆模型

- LSTM (Long Short-Term Memory networks)
  - Forget gate（遗忘门）：记住前一时刻单元状态的多少内容

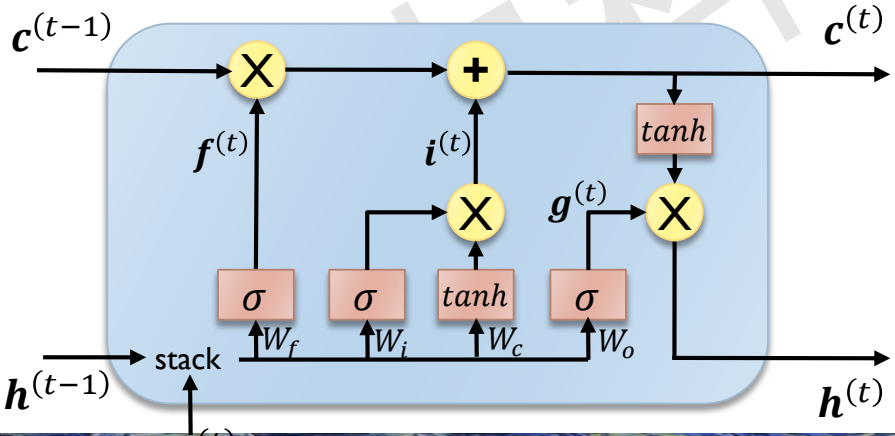
$$\boldsymbol{f}^{(t)} = \sigma\left(U^f \boldsymbol{x}^{(t)} + W^f \boldsymbol{h}^{(t-1)} + \boldsymbol{b}^f\right) = \sigma\left(\mathcal{W}_f \begin{pmatrix} \boldsymbol{h}^{(t-1)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}\right)$$

  - Input gate（输入门）：写入多少输入到当前单元

$$\boldsymbol{i}^{(t)} = \sigma\left(U^i \boldsymbol{x}^{(t)} + W^i \boldsymbol{h}^{(t-1)} + \boldsymbol{b}^i\right) = \sigma\left(\mathcal{W}_i \begin{pmatrix} \boldsymbol{h}^{(t-1)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}\right)$$

  - Output gate（输出门）：输出多少当前单元状态

$$\boldsymbol{g}^{(t)} = \sigma\left(U^o \boldsymbol{x}^{(t)} + W^o \boldsymbol{h}^{(t-1)} + \boldsymbol{b}^o\right) = \sigma\left(\mathcal{W}_o \begin{pmatrix} \boldsymbol{h}^{(t)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}\right)$$



内部状态更新：

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \tilde{\boldsymbol{c}}^{(t)}$$

$$\tilde{\boldsymbol{c}}^{(t)} = tanh\left(U\boldsymbol{x}^{(t)} + W\boldsymbol{h}^{(t-1)} + \boldsymbol{b}\right) = \sigma\left(\mathcal{W}_c \begin{pmatrix} \boldsymbol{h}^{(t-1)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}\right)$$

LSTM cell输出：

$$\boldsymbol{h}^{(t)} = \boldsymbol{g}^{(t)} tanh(\boldsymbol{c}^{(t)})$$

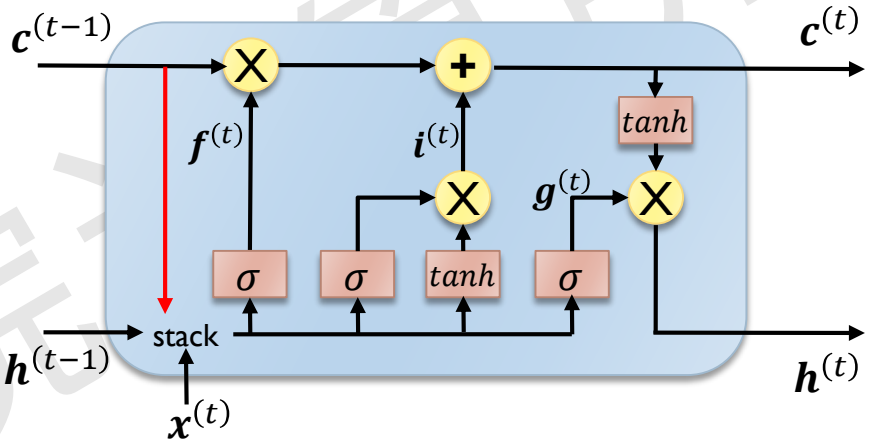
长短期记忆模型

- LSTM变体
  - 窥视孔连接(peephole connection)：门值不仅取决于h_{t-1}和x_t，也取决于上一个单元状态的值C_{t-1}；

$$f^{(t)} = \sigma\left(U^f x^{(t)} + W^f h^{(t-1)} + M^f c^{(t-1)} + b^f\right)$$

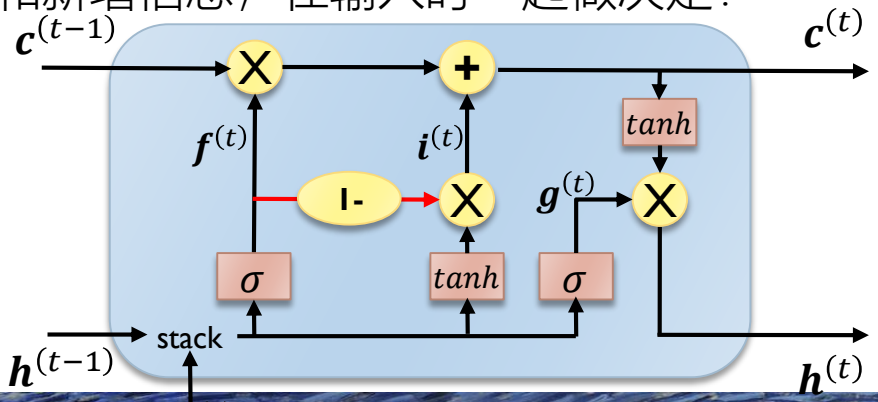
$$i^{(t)} = \sigma\left(U^i x^{(t)} + W^i h^{(t-1)} + M^i c^{(t-1)} + b^i\right)$$

$$g^{(t)} = \sigma\left(U^o x^{(t)} + W^o h^{(t-1)} + M^o c^{(t-1)} + b^o\right)$$



  - 耦合输入门和遗忘门：不单独决定遗忘和新增信息，在输入时一起做决定：

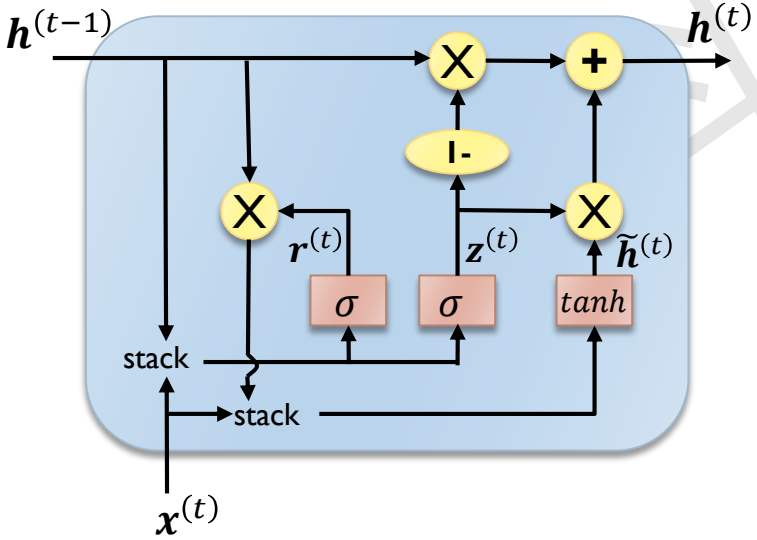
$$c^{(t)} = f^{(t)} c^{(t-1)} + (1 - f^{(t)})\tilde{c}^{(t)}$$



长短期记忆模型

- GRU (Gated Recurrent Unit)

  - Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.
  - 在LSTM的基础上，将单元状态和隐藏状态合并，将遗忘门和输入门合并为更新门，无输出门。更新门决定历史信息和当前信息如何相加；重置门决定保留多少历史信息进入当前信息；



更新门（update gate）：

$$z^{(t)} = \sigma\left(U^z x^{(t)} + W^z h^{(t-1)} + b^z\right)$$

重置门（reset gate）：

$$r^{(t)} = \sigma\left(U^r x^{(t)} + W^r h^{(t-1)} + b^r\right)$$

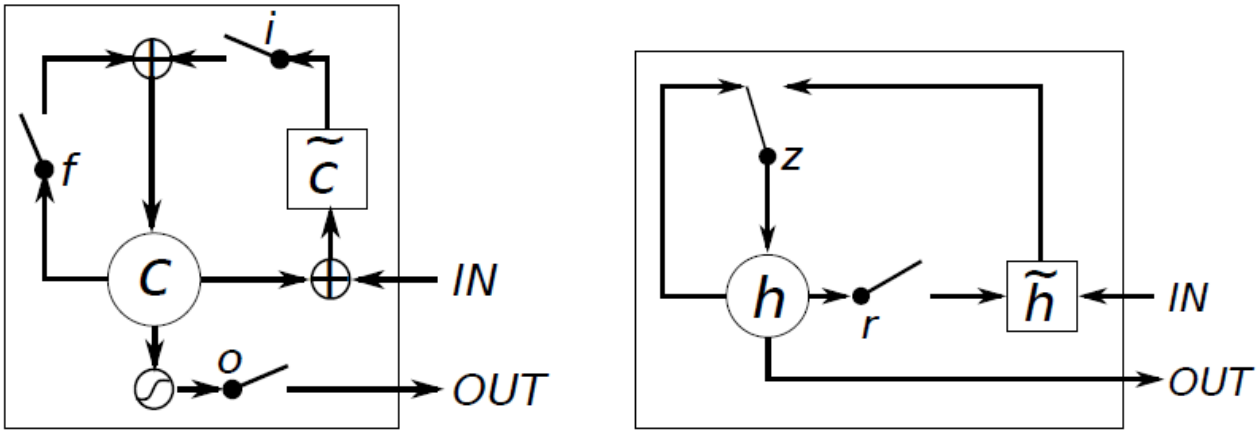
隐藏状态更新：

$$h^{(t)} = (1 - z^{(t)})h^{(t-1)} + z^{(t)}\tilde{h}^{(t)}$$

$$\tilde{h}^{(t)} = tanh\left(Ux^{(t)} + W(r^{(t)}h^{(t-1)}) + b\right)$$

长短期记忆模型

- LSTM与GRU



(a) Long Short-Term Memory　　(b) Gated Recurrent Unit

- 哪个模型更好无定论。GRU参数量更少，训练速度快；在训练数据足够的情况下，LSTM的表征能力更强；可自行选择；

提纲

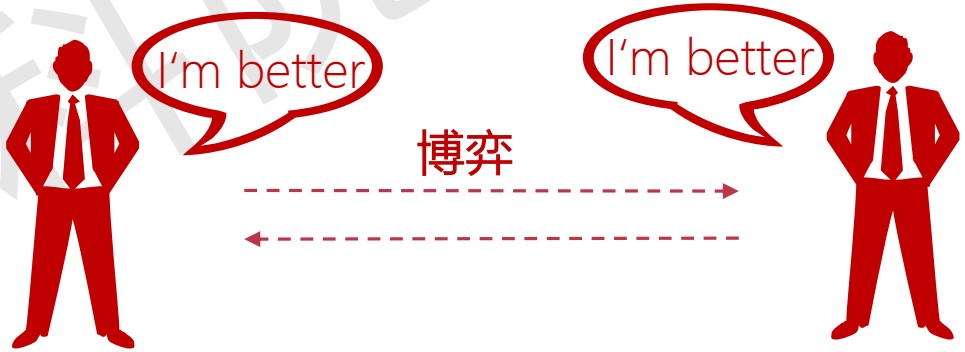
- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

神经网络的无监督学习

- 有监督学习

  - 损失函数是给定的，网络学习到的是数据的模式/特征；

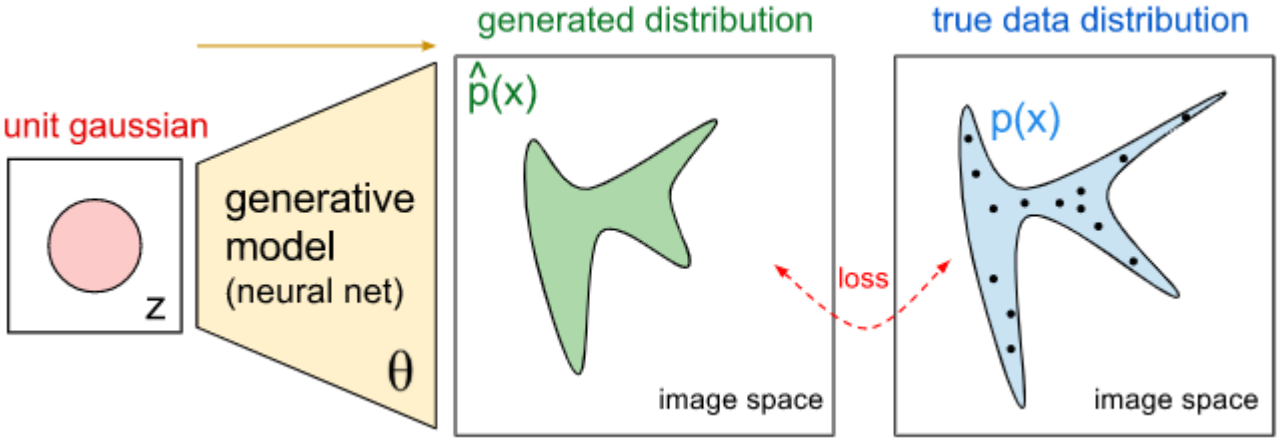
- 无监督学习



  - 损失函数学习得到，网络学习到的是数据的分布；

生成对抗网络GAN

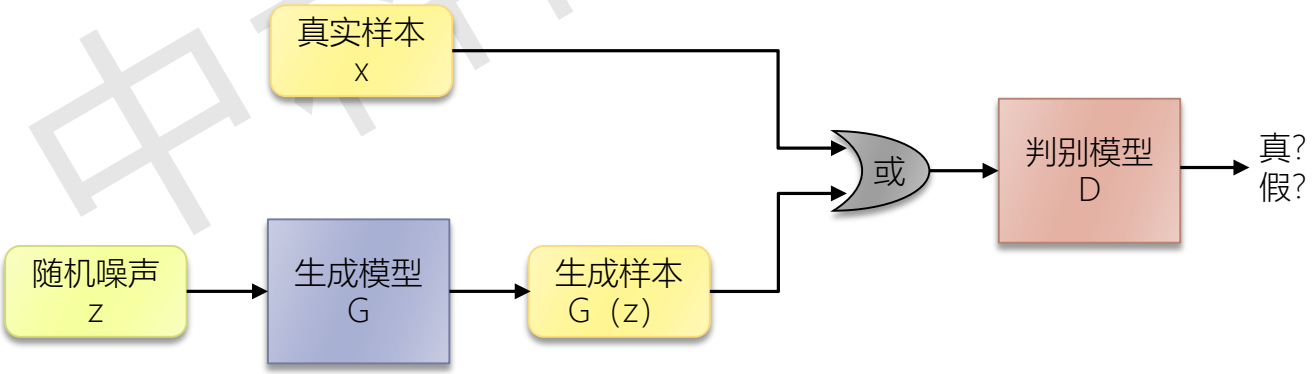
- 2014年Ian Goodfellow提出生成式对抗网络GAN
  - Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]// International Conference on Neural Information Processing Systems. 2014.
  - 20年来机器学习领域最酷的想法——Yann LeCun
- 解决的问题：从训练样本中学习出新样本。为无监督、预测学习提供算法框架。
- 和苏格拉底的辩证法有相似之处



GAN学习到了真实样本集的数据分布

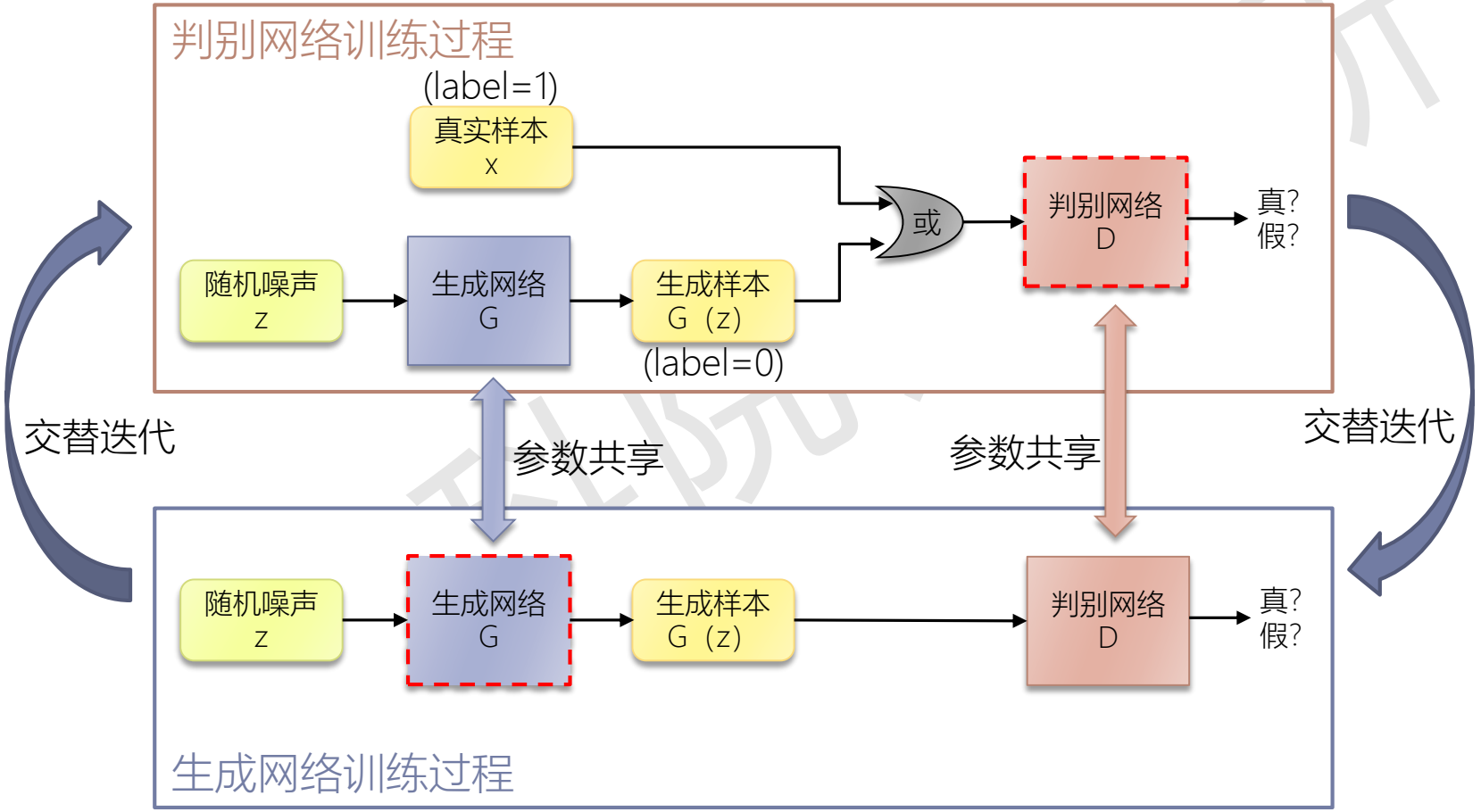
生成对抗网络GAN

- 模型由两部分组成
  - **生成器**（伪装者）：找出观测数据内部的统计规律，尽可能生成能够以假乱真的样本，使判别网络输出接近0.5，难以区分真假。
  - **判别器**（警察）：判断输入数据是来自真实样本集还是生成样本集。如果输入是真样本，输出接近1；如果输入是生成样本，输出接近0。



GAN训练过程

- 更新判别网络权值参数，使其能分清真假样本。



- 更新生成网络权值参数，使其生成的假样本被判别网络识别为真样本。

GAN训练过程

- 优化判别器D

$$J^{(D)} = -\mathrm{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] - \mathrm{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

  - 优化目标：输入真样本x时，输出接近1；输入生成样本G(z)时，输出接近0。

- 优化生成器G

$$J^{(G)} = \mathrm{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

  - 优化目标：生成的假样本G(z)，被判别器判断为接近1，即 (1-D(G(z)) 越小越好。

- GAN训练是极小极大博弈问题（或零和博弈）

$$\min_G \max_D V(D,G) = -J^{(D)} = \mathrm{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathrm{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

生成对抗网络GAN

- GAN有一个整体的损失函数，而不需要对生成网络和对抗网络分别指定具体的损失函数，有较强的通用性。

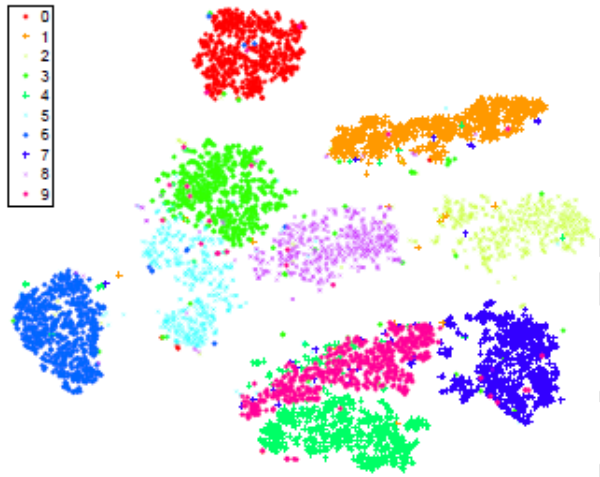
- 问题
  - 梯度消失：当判别器以高置信度成功判断生成器生成的样本为假样本时，生成器的梯度会消失。训练早期，$\log(1-D(G(z)))$ 会饱和
    - 应对方法：修改生成器的代价函数

$$J^{(G)} = -\mathrm{E}_{z\sim p_z(z)}[\log(D(G(z)))]$$

  - 模式崩溃：生成器只生成几种模式的样本，生成样本缺乏多样性

模式崩溃(model collapse)

- 产生原因： GAN的损失函数使判别器假样本的惩罚是巨大的，一旦生成的某一类假样本成功骗过判别器，生成器就趋向于生成相似的样本。



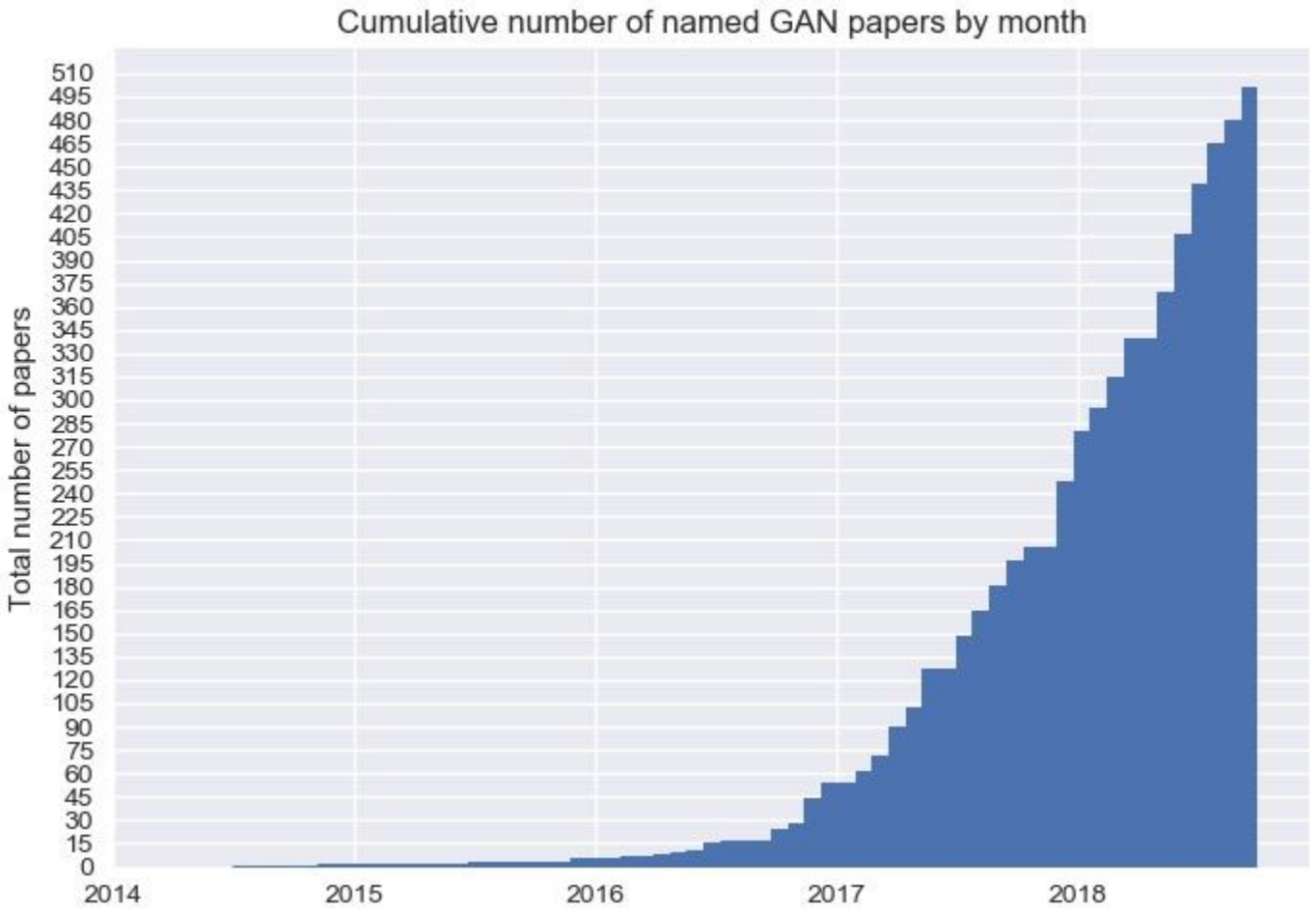
MNIST数据集的t-SNE图，10个团簇对应10种模式。

模式崩溃时，只生成其中的几种模式，生成样本缺乏多样性。

van der Maaten, L. & Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research,* **2008**, 2579-2605

- 应对方法（WGAN）：采用更加平滑的损失函数，参见Wasserstein GAN

GAN相关研究



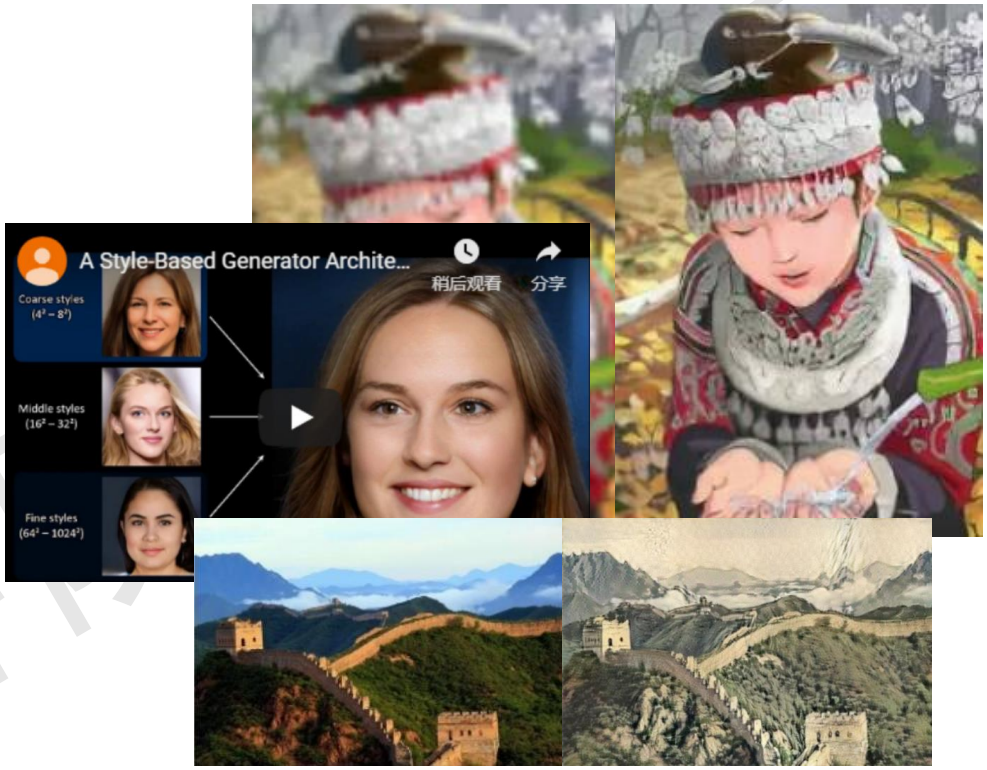
https://github.com/hindupuravinash/the-gan-zoo

GAN结构

- 卷积GAN
  - DCGAN：将GAN中全连接神经网络扩展到卷积神经网络
  - ResGAN：图像恢复，ResNet
  - SRGAN：超分辨率，ResNet
  - CycleGAN：图像转换
- 条件GAN
  - CGAN
  - InfoGAN
- 集成推断模型的GAN
  - BiGAN
- 对抗自编码器
  - VAE-GAN

GAN应用

- 人脸生成
- 风格转换
- Super resolution
- …



GAN zoo：https://deephunt.in/the-gan-zoo-79597dc8c347
GAN代码合集：https://github.com/zhangqianhui/AdversarialNetsPapers
GAN应用合集：https://github.com/nashory/gans-awesome-applications

提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

Driving Example

像艺术家一样

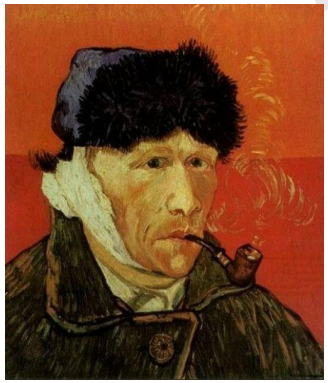
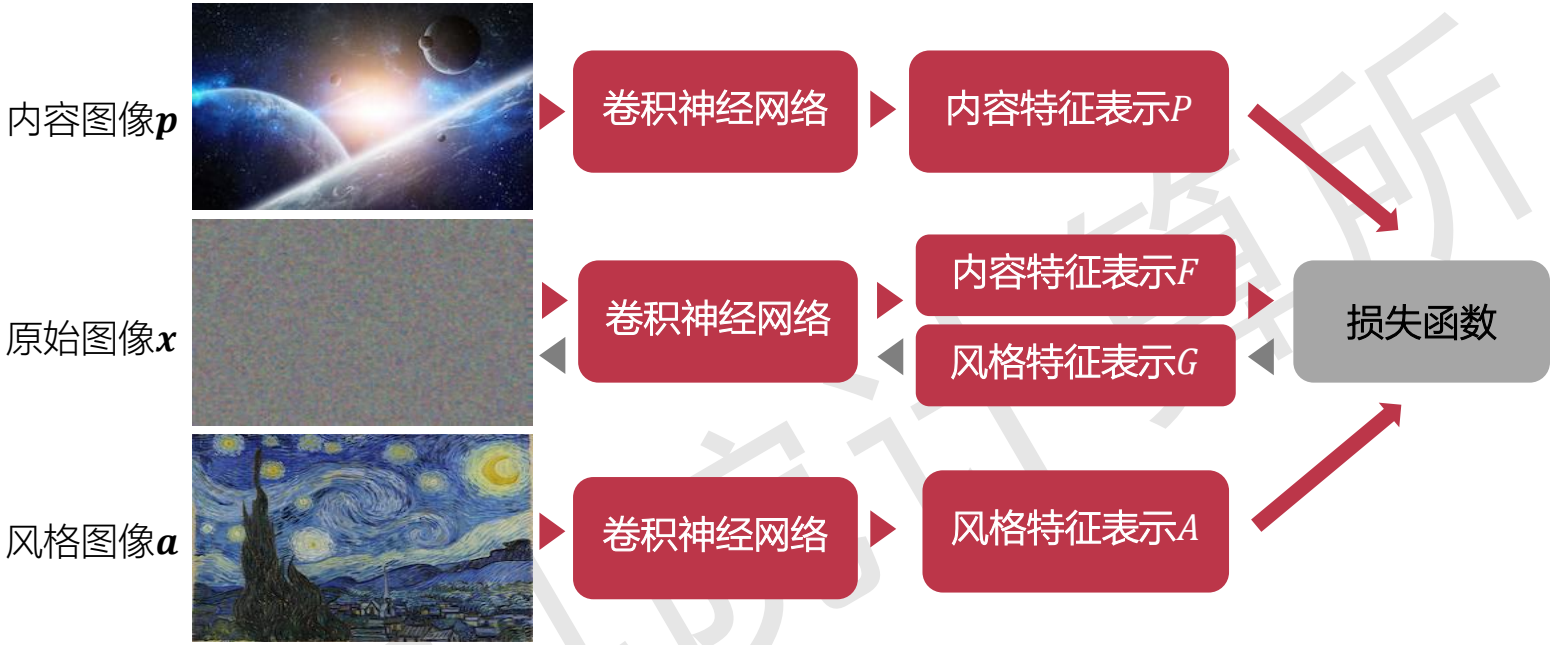


Image style transfer

- Gatys et al. Image Style Transfer Using Convolutional Neural Networks[C]. 2016.

Image style transfer



- 给定一张风格图像 a 和一张内容图像 p；
- 风格图像经过 CNN 生成的 feature maps 组成风格特征集A；内容图像 p 通过 CNN 生成的 feature maps 组成内容特征集P；
- 输入一张随机噪声图像 x，随机噪声图像 x 通过 CNN 生成的 feature maps 构成内容特征和风格特征集合 F和 G ， 目标损失函数由 A, P, F ,G 计算得到；
- 优化函数是希望调整图像 x，使其最后看起来既保持内容图像 p的内容, 又有风格图像 a 的风格。

Image style transfer

- 论文中使用的CNN网络为在imageNet上训练好的VGG19，去除了最后的全连接层和softmax。
- 内容损失函数：
  - 只取conv4单层特征来计算内容损失；
  - 计算内容图片特征和噪声图片特征之间的欧式距离；

$$L_{content}(\boldsymbol{p}, \boldsymbol{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

- l: 用于计算内容特征的层数.
- F_{ij}^{l}: 生成图片在第l层第i个特征图上位置j处的特征值
- P_{ij}^{l}: 内容图片在第l层第i个特征图上位置j处的特征值
- p: 内容图片
- x: 生成图片

Image style transfer

- 风格损失函数：
  - 取 conv1~conv5共5层的特征来计算风格损失；
  - 使用相关矩阵来表示图像的风格；

$$L_{style}(\boldsymbol{a}, \boldsymbol{x}) = \sum_{l} w_l E_l$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

(gram矩阵)

- L: 用于计算风格特征的层数
- w_l: 第l层的E_l用于计算风格损失的权重，文中都取0.2
- a: 初始风格图片
- x: 生成图片
- A_{ij}^l: 风格图片在l层第i个特征图和第j个特征图的内积
- G_{ij}^l: 生成图片在l层第i个特征图和第j个特征图的内积
- M_l: 第l层的输出特征图的大小
- N_l: 第l层的输出特征图的数目
- F_{ik}^l: 第l层第i个特征图上位置k处的特征值

Image style transfer

$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x})$$

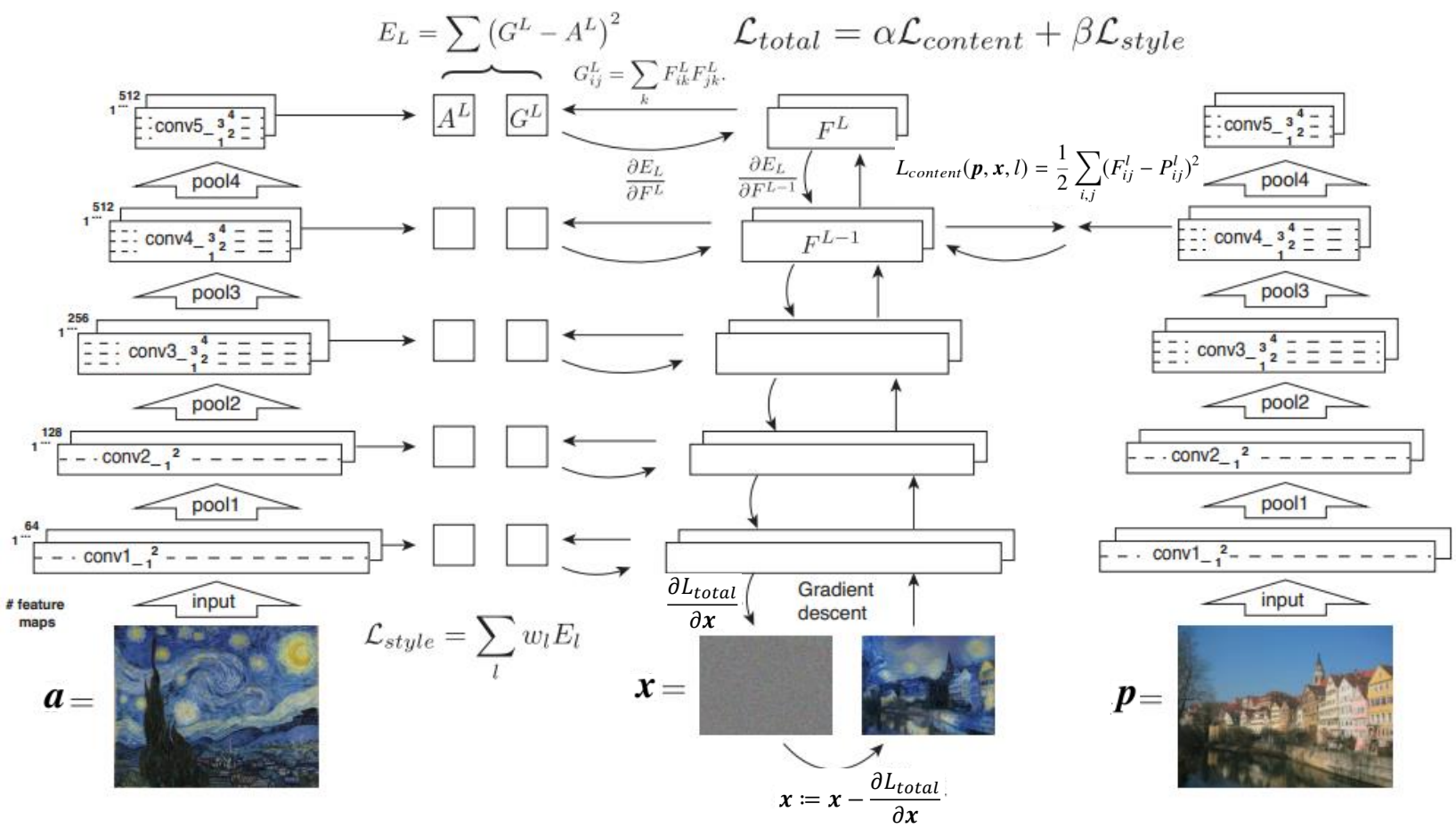
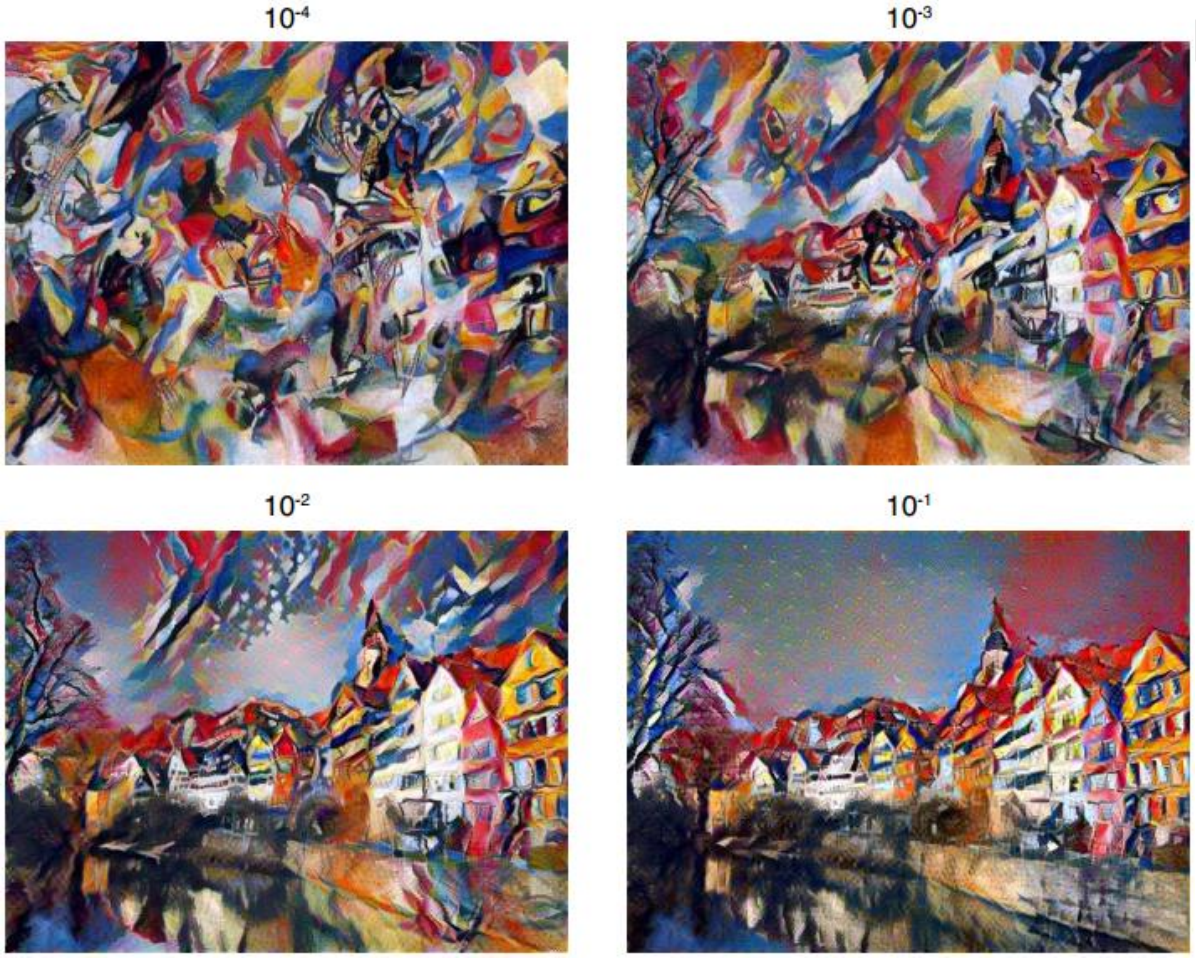


Image style transfer

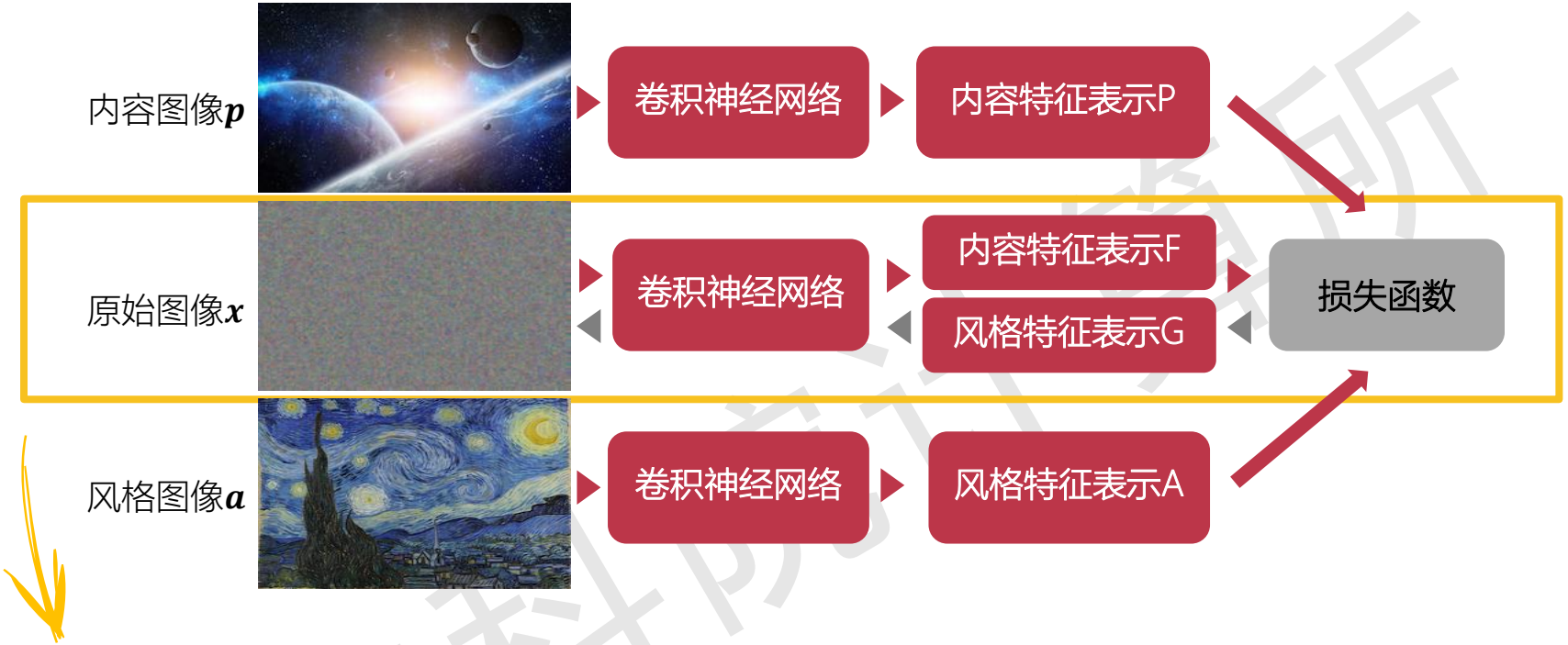
$$L_{total}(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{x}) = \alpha L_{content}(\boldsymbol{p}, \boldsymbol{x}) + \beta L_{style}(\boldsymbol{a}, \boldsymbol{x})$$

α/β 越大

内容越具象



Real-Time Image style transfer



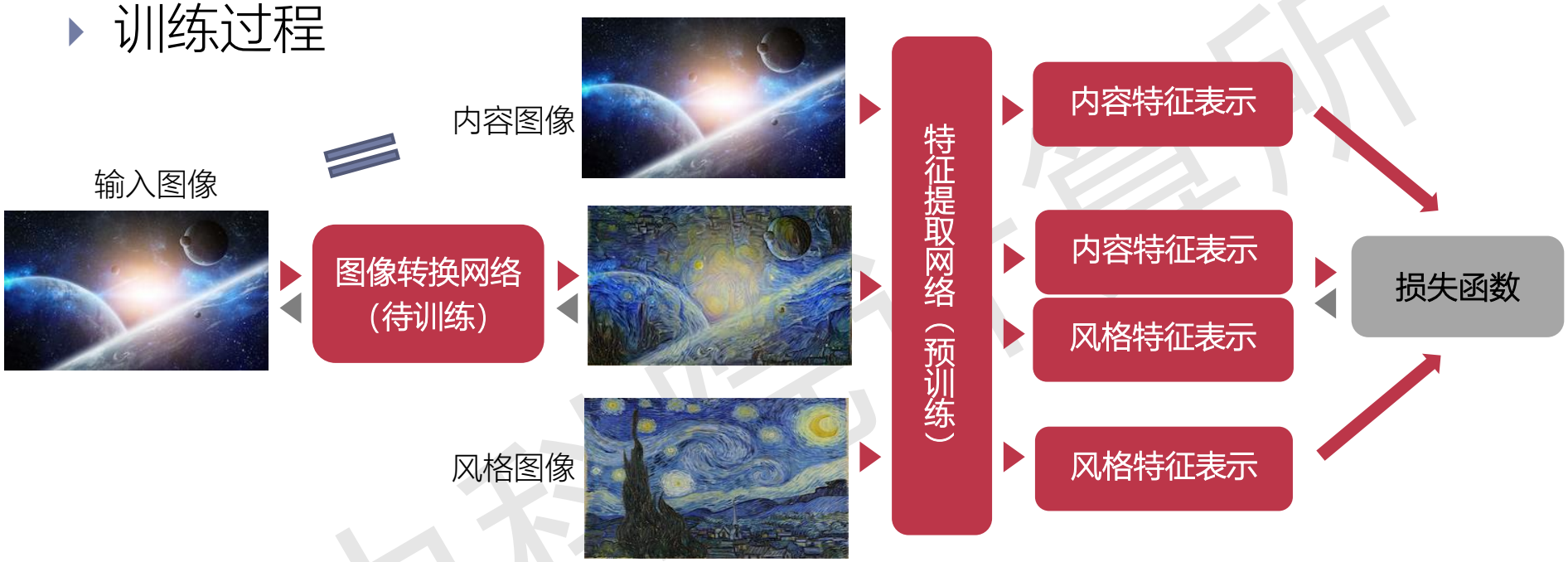
对每一张待转换图片，都要进行前馈和反馈调优过程，无法做到实时转换；

能够把图像生成网络提前训练好，

使风格转换过程不需要进行反向训练？

Real-Time Image style transfer

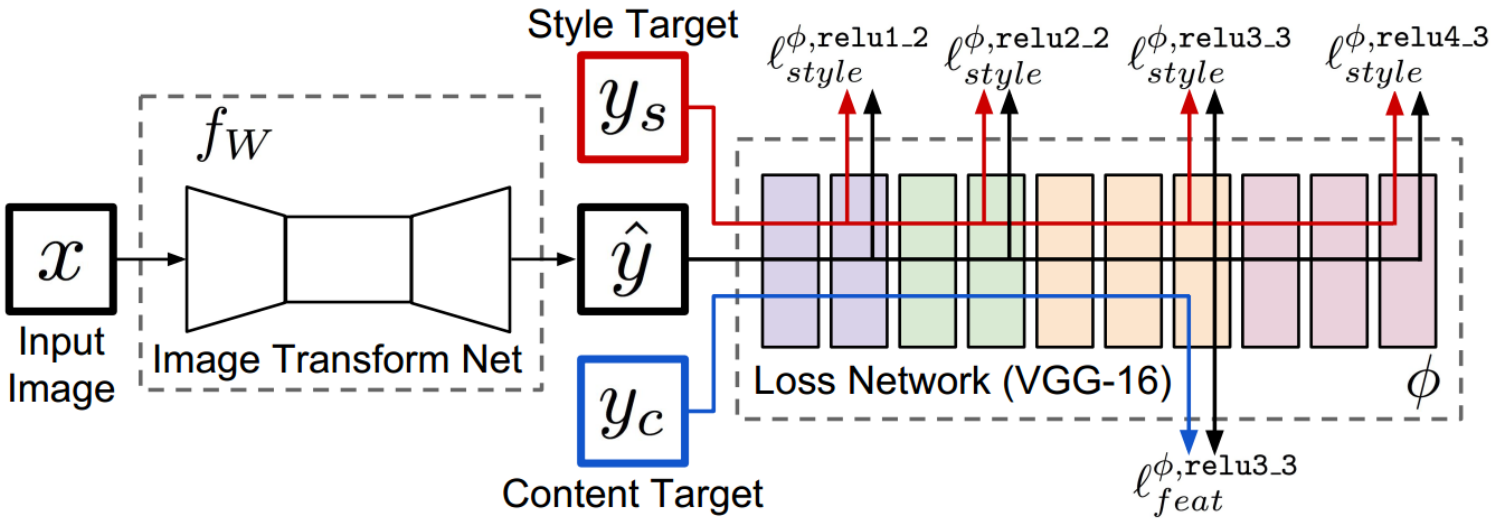
- 训练过程



- 实时转换过程

Real-Time Image style transfer

- Johnson J, Alahi A, Feifei L, et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution[J]. 2016.
- 项目实现：https://github.com/jcjohnson/fast-neural-style



- Image Transform Net
  - 深度卷积网络，参考DCGAN的设计思想：用步幅卷积替代pooling、每个卷积层后接BatchNorm和Relu；
  - 增加残差结构，使网络更易训练；

Real-Time Image style transfer

Content Gatys et al. Ours

Image Size	Gatys *et al.* [11] 100	300	500	Ours	Speedup 100	300	500
256 × 256	3.17	9.52s	15.86s	**0.015s**	212x	636x	**1060x**
512 × 512	10.97	32.91s	54.85s	**0.05s**	205x	615x	**1026x**
1024 × 1024	42.89	128.66s	214.44s	**0.21s**	208x	625x	**1042x**

提纲

- 适合图像处理的卷积神经网络
- 基于CNN的图像分类算法
- 基于CNN的图像检测算法
- 序列模型：循环神经网络
- 序列模型：长短期记忆模型
- 生成对抗网络GAN
- Driving Example
- 小结

小结

1. 适合图像处理的卷积神经网络

包括卷积层，池化层，归一化层，全连接层，softmax 计算原理及层的排布规律；

2. 基于CNN的图像分类神经网络算法

主要介绍了AlexNet(LRN，dropout)，VGG，Inception系列(1*1卷积，BatchNorm)，ResNet 经典的神经网路分类算法；

3. 基于CNN的图像检测神经网络算法

主要介绍了检测算法评价指标(IoU，mAP)，R-CNN系列(Region Proposal，NMS， ROIPooling，RPN，anchor box)，YOLO，SSD；

4. 循环神经网络

主要介绍了循环神经网络的应用，网络结构，正/反向过程，以及梯度消失产生原因；

5. 长短期记忆模型(LSTM/GRU)

6. 生成对抗网络GAN

包括原始GAN结构，模式崩溃，DCGAN，以及conditionalGAN；

7. Driving Example

课程教师

陈云霁 研究员 cyj@ict.ac.cn

李玲 研究员 liling@iscas.ac.cn

李威 副研究员 liwei2017@ict.ac.cn

杜子东 副研究员 duzidong@ict.ac.cn

谢谢大家！